















Author

Patrick J. [O'Connell]

Title

Application of Fast Fourier Transform and Spectra Fitting Programs

to an Automated Seismic Processor (ASP) for Microearthquake Networks

RESEARCH PROJECT

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, in partial satisfaction of the requirements for the degree of Master of Science, Plan II.

Approval for the Report and Comprehensive Examination:

Committee: \_\_\_\_\_, Research Adviser

\_\_\_\_\_  
Date

\_\_\_\_\_  
Date

5551-1071

## ABSTRACT

This report explains the operation of a variable length FFT Algorithm and an associated Spectra FIT Program for use in an Automated Seismic Processor (ASP) System for Microearthquake Networks. Sufficient background is given concerning the requirement for the remote sensing and processing being undertaken. The interaction of these algorithms with the ASP software system is explained. The algorithm and its implementation on the RCA 1802 CMOS microprocessor is described in detail. Debugging tools which were developed for the task are explained. Results of the operation, both for tests of the FFT operation and for simulated earthquake operation, are presented. The Program Code, Run Time Code, FFT Validation Results and Earthquake Results are included.





## ACKNOWLEDGEMENT

The IET Project Development could not have been accomplished without the assistance of a number of individuals. Mr. Tom McElvilly, professor of geology and Geophysics at the University of California at Berkeley and State Geologist of California Geological Survey, with his Division, provided direction throughout this effort. Jerry Newman and Jim Katschke performed the initial system development. Edricine

### DEDICATION

To my Mother, and in memory of my Father.

Barrell Hall, John Friday, and Martin Miller of the Seismographic Station offered ongoing support throughout the development. In addition Mr. Alvin H. Duglin, professor of Computer Science at the University of California at Berkeley, offered direction and assistance during IET/MS project development and during the writing of this thesis. This work was supported by The National Science Foundation for Conservation and Renewable Energy, Division of Geosciences and Atmospheric Sciences, and by the Assistant Secretary for Nuclear Waste Management, U.S. Department of Energy, under contract W-7405-ENG-82.



### ACKNOWLEDGEMENT

The ASP FFT/FIT development could not have been accomplished without the assistance of a number of individuals. Dr. Tom McEvelly, professor of Geology and Geophysics at the University of California at Berkeley and Ernie Mayer of Lawrence Berkeley Laboratory, Earth Sciences Division, provided direction throughout this effort. Jerry Heinsen and Jim Bartschi performed the initial system definition, fabrication, software development, and overall debugging. Russell Sell, John Friday, and Marvin Hilger of the Seismographic Station offered ongoing support throughout the development. In addition Dr. Alvin M. Despain, professor of Computer Science at the University of California at Berkeley, offered direction and assistance during FFT/FIT program development and during the writing of this thesis. This work was supported by the Assistant Secretary for Conservation and Renewable Energy, Division of Geothermal and Hydropower Technologies, and by the Assistant Secretary for Nuclear Waste Isolation, U.S. Department of Energy, under Contract W-7405-ENG-48.

## CONCLUSIONS

The first conclusion is that the present state of knowledge in the field of the development of the human mind is very limited. It is not possible to give a complete answer to the question of the development of the human mind. The second conclusion is that the development of the human mind is a process which is influenced by many factors. The third conclusion is that the development of the human mind is a process which is influenced by many factors. The fourth conclusion is that the development of the human mind is a process which is influenced by many factors. The fifth conclusion is that the development of the human mind is a process which is influenced by many factors. The sixth conclusion is that the development of the human mind is a process which is influenced by many factors. The seventh conclusion is that the development of the human mind is a process which is influenced by many factors. The eighth conclusion is that the development of the human mind is a process which is influenced by many factors. The ninth conclusion is that the development of the human mind is a process which is influenced by many factors. The tenth conclusion is that the development of the human mind is a process which is influenced by many factors.



TABLE OF CONTENTS

TABLE OF CONTENTS . . . . .	vii
LIST OF FIGURES . . . . .	xv
1.0 INTRODUCTION . . . . .	1
1.1 Objective . . . . .	1
1.2 Scope . . . . .	2
1.3 Background . . . . .	2
2.0 ASP OPERATION . . . . .	5
2.1 Earthquake Dynamics . . . . .	6
2.2 System Operation . . . . .	6
2.2.1 ASP Software . . . . .	7
2.2.1.1 WORKER Routines . . . . .	7
2.2.1.2 BOSS Routines . . . . .	11
2.2.2 ASP Hardware . . . . .	12
2.2.3 ASP Operation . . . . .	15
3.0 FFT/FIT PROGRAM DESIGN . . . . .	17
3.1 WORKER Operation . . . . .	18
3.2 FFT/FIT Requirements . . . . .	18
3.3 Mathematical Precision . . . . .	19
3.4 RCA 1802 Architecture Considerations . . . . .	20
3.5 Development System Support . . . . .	23
3.6 Programming Techniques . . . . .	25
3.6.1 Structured Programming . . . . .	25
3.6.2 Numbering Notations . . . . .	26
3.6.3 Programming Conventions . . . . .	27
3.7 Human Factors . . . . .	28
4.0 DATA STRUCTURES . . . . .	30
4.1 Register Names . . . . .	30
4.2 Memory Allocations . . . . .	30
4.3 Routine Locations . . . . .	33
4.3.1 Worker Routines . . . . .	33
4.3.2 FFT Routines . . . . .	33
4.3.3 FIT Routines . . . . .	34
4.3.4 Run Time Routines . . . . .	34
4.4 Stacks, Buffers and FIFOs . . . . .	35
4.4.1 Stacks . . . . .	35
4.4.2 Buffers . . . . .	35
4.4.3 FIFOs . . . . .	36
4.5 Variables . . . . .	37
4.5.1 WORKER Variables . . . . .	37
4.5.2 Global Variables . . . . .	37
4.5.2.1 FFT/FIT Global Variables . . . . .	37
4.5.2.2 FIT Global Variables . . . . .	39

ASP FFT Program  
Table of Contents

4.5.3	Local Variables . . . . .	39
4.6	Constants . . . . .	40
4.6.1	FFT/FIT Constants . . . . .	40
4.7	Tables . . . . .	42
4.7.1	FFT/FIT Tables . . . . .	42
4.7.1.1	TRGTBL (Trig Table) . . . . .	42
4.7.1.2	LOGTBL (Logarithm Table) . . . . .	42
4.7.1.3	INSFIX (Instrument Correction Table) . . . . .	42
4.7.2	Run Time Tables . . . . .	43
5.0	FFT/FIT PROGRAM DESCRIPTION . . . . .	44
5.0.1	FFTMN (FFT Main) . . . . .	44
5.0.1.1	Function . . . . .	44
5.0.1.2	Calling Program . . . . .	44
5.0.1.3	Inputs . . . . .	44
5.0.1.4	Internal Variables . . . . .	44
5.0.1.5	Subroutines Used . . . . .	45
5.0.1.6	Output . . . . .	45
5.0.1.7	Description . . . . .	45
5.1	FFT Programs . . . . .	47
5.1.1	LNSET (Length Set) . . . . .	47
5.1.1.1	Function . . . . .	47
5.1.1.2	Calling Program . . . . .	47
5.1.1.3	Inputs . . . . .	47
5.1.1.4	Internal Variables . . . . .	48
5.1.1.5	Subroutines Used . . . . .	48
5.1.1.6	Outputs . . . . .	48
5.1.1.7	Description . . . . .	48
5.1.2	SCALE . . . . .	50
5.1.2.1	Function . . . . .	50
5.1.2.2	Calling Program . . . . .	50
5.1.2.3	Inputs . . . . .	50
5.1.2.4	Internal Variables . . . . .	50
5.1.2.5	Subroutines Used . . . . .	50
5.1.2.6	Output . . . . .	50
5.1.2.7	Description . . . . .	51
5.1.3	SHAPE . . . . .	52
5.1.3.1	Function . . . . .	52
5.1.3.2	Calling Program . . . . .	52
5.1.3.3	Inputs . . . . .	52
5.1.3.4	Internal Variables . . . . .	53
5.1.3.5	Subroutines Used . . . . .	53
5.1.3.6	Output . . . . .	53
5.1.3.7	Description . . . . .	54
5.1.4	FFT . . . . .	56
5.1.4.1	Function . . . . .	56
5.1.4.2	Calling Program . . . . .	57
5.1.4.3	Inputs . . . . .	57
5.1.4.4	Internal Variables . . . . .	57
5.1.4.5	Subroutines Used . . . . .	59
5.1.4.6	Output . . . . .	59
5.1.4.7	Description . . . . .	59

5.1.5	UNSCRM (Unscramble Data)	61
5.1.5.1	Function	61
5.1.5.2	Calling Program	61
5.1.5.3	Inputs	61
5.1.5.4	Internal Variables	61
5.1.5.5	Subroutines Used	62
5.1.5.6	Output	62
5.1.5.7	Description	62
5.1.6	ORDER (Column Reorder)	64
5.1.6.1	Function	64
5.1.6.2	Calling Program	64
5.1.6.3	Inputs	64
5.1.6.4	Internal Variables	65
5.1.6.5	Subroutines Used	65
5.1.6.6	Output	65
5.1.6.7	Description	66
5.2	FIT Programs	71
5.2.1	MAGAPX (Magnitude Approximate)	73
5.2.1.1	Function	73
5.2.1.2	Calling Program	74
5.2.1.3	Inputs	74
5.2.1.4	Internal Variables	74
5.2.1.5	Subroutines Used	74
5.2.1.6	Output	74
5.2.1.7	Description	75
5.2.2	SMTGMX (Smooth and Tag Maximum)	77
5.2.2.1	Function	77
5.2.2.2	Calling Program	77
5.2.2.3	Inputs	77
5.2.2.4	Internal Variables	78
5.2.2.5	Subroutines Used	78
5.2.2.6	Output	78
5.2.2.7	Description	79
5.2.3	CKFMAX (Check Maximum Value)	82
5.2.3.1	Function	82
5.2.3.2	Calling Program	82
5.2.3.3	Inputs	82
5.2.3.4	Internal Variables	83
5.2.3.5	Subroutines Used	83
5.2.3.6	Output	83
5.2.3.7	Description	83
5.2.4	GMACPT (GAMMA Compute)	87
5.2.4.1	Function	87
5.2.4.2	Calling Program	87
5.2.4.3	Inputs	87
5.2.4.4	Internal Variables	88
5.2.4.5	Subroutines Used	89
5.2.4.6	Output	89
5.2.4.7	Description	90
5.2.5	FOLPL (Compute FO and Long Period Level)	94
5.2.5.1	Function	94
5.2.5.2	Calling Program	94

ASP FFT Program  
Table of Contents

5.2.5.3	Inputs . . . . .	94
5.2.5.4	Internal Variables . . . . .	95
5.2.5.5	Subroutines Used . . . . .	95
5.2.5.6	Output . . . . .	95
5.2.5.7	Description . . . . .	95
5.3	Subroutines . . . . .	97
5.3.1	WRPTST (Wraparound Test) . . . . .	97
5.3.1.1	Function . . . . .	97
5.3.1.2	Calling Programs . . . . .	97
5.3.1.3	Inputs . . . . .	97
5.3.1.4	Internal Variables . . . . .	98
5.3.1.5	Subroutines Used . . . . .	98
5.3.1.6	Output . . . . .	98
5.3.1.7	Description . . . . .	98
5.3.2	CSMULT and SNMULT (Cosine and Sine Multiply) . . . . .	99
5.3.2.1	Function . . . . .	99
5.3.2.2	Calling Programs . . . . .	99
5.3.2.3	Inputs . . . . .	99
5.3.2.4	Internal Variables . . . . .	100
5.3.2.5	Subroutines Used . . . . .	100
5.3.2.6	Output . . . . .	100
5.3.2.7	Description . . . . .	101
5.3.3	ADDSTF (Add and Stuff) . . . . .	102
5.3.3.1	Function . . . . .	102
5.3.3.2	Calling Programs . . . . .	102
5.3.3.3	Inputs . . . . .	103
5.3.3.4	Internal Variables . . . . .	103
5.3.3.5	Subroutines Used . . . . .	103
5.3.3.6	Output . . . . .	103
5.3.3.7	Description . . . . .	103
5.3.4	NEWVAL (New Value) . . . . .	104
5.3.4.1	Function . . . . .	104
5.3.4.2	Calling Programs . . . . .	104
5.3.4.3	Inputs . . . . .	104
5.3.4.4	Internal Variables . . . . .	105
5.3.4.5	Subroutines Used . . . . .	105
5.3.4.6	Output . . . . .	105
5.3.4.7	Description . . . . .	105
5.3.5	TRGSET and SHPSET (Trig and Shape Set) . . . . .	106
5.3.5.1	Function . . . . .	106
5.3.5.2	Calling Programs . . . . .	107
5.3.5.3	Inputs . . . . .	107
5.3.5.4	Internal Variables . . . . .	107
5.3.5.5	Subroutines Used . . . . .	108
5.3.5.6	Output . . . . .	108
5.3.5.7	Description . . . . .	108
5.3.6	DTASCL (Data Scale) . . . . .	109
5.3.6.1	Function . . . . .	109
5.3.6.2	Calling Programs . . . . .	110
5.3.6.3	Inputs . . . . .	110
5.3.6.4	Internal Variables . . . . .	110
5.3.6.5	Subroutines Used . . . . .	110



5.3.6.6	Output . . . . .	110
5.3.6.7	Description . . . . .	111
5.3.7	BDRYST/CK (Boundary Set and Check) . . . . .	114
5.3.7.1	Function . . . . .	114
5.3.7.2	Calling Programs . . . . .	114
5.3.7.3	Inputs . . . . .	114
5.3.7.4	Internal Variables . . . . .	115
5.3.7.5	Subroutines Used . . . . .	115
5.3.7.6	Output . . . . .	115
5.3.7.7	Description . . . . .	115
6.0	RUN TIME PROGRAMS . . . . .	117
6.1	Operation . . . . .	117
6.1.1	Purpose . . . . .	117
6.1.2	Data Structures . . . . .	118
6.1.2.1	Memory Locations . . . . .	118
6.1.2.2	Routine Locations . . . . .	119
6.1.2.3	Variables . . . . .	121
6.1.2.4	Tables . . . . .	122
6.2	SYSRUN (System Run) . . . . .	123
6.2.1	Function . . . . .	123
6.2.2	Calling Program . . . . .	124
6.2.3	Inputs . . . . .	124
6.2.4	Internal Variables . . . . .	124
6.2.5	Subroutines Used . . . . .	124
6.2.6	Output . . . . .	124
6.2.7	Description . . . . .	125
6.3	LDFIFO (Load FIFO) . . . . .	127
6.3.1	Function . . . . .	127
6.3.2	Calling Program . . . . .	127
6.3.3	Inputs . . . . .	127
6.3.4	Internal Variables . . . . .	127
6.3.5	Subroutines Used . . . . .	127
6.3.6	Output . . . . .	128
6.3.7	Description . . . . .	128
6.4	Data Input . . . . .	130
6.5	Trace Programs . . . . .	131
6.5.1	FFT Complete Trace . . . . .	131
6.5.2	FFT Out Trace . . . . .	132
6.5.3	FFT No Trace . . . . .	132
6.5.4	Multi FFT . . . . .	132
6.5.5	8085 Full Trace . . . . .	132
6.6	Problems . . . . .	134
7.0	PROGRAM VALIDATION . . . . .	135
7.1	Subprogram Validation Series . . . . .	135
7.2	Input/Output Validation Series . . . . .	136
7.3	Frequency Input . . . . .	136
8.0	OPERATIONAL RESULTS . . . . .	137
8.1	FFT Processing Times . . . . .	137
8.2	ASP System Results . . . . .	137



ASP FFT Program  
Table of Contents

9.0	SUMMARY . . . . .	140
9.1	Results . . . . .	140
9.2	Conclusion . . . . .	141
10.0	REFERENCES . . . . .	142

# APPENDIX A PROGRAM CODE

A.1	PROGRAM HEADER . . . . .	A-1
A.2	FFT/FIT CODE . . . . .	A-7
A.2.0.1	FFTMN (FFT Main) . . . . .	A-7
A.2.1	FFT Code . . . . .	A-12
A.2.1.1	LNSET (Length Set) . . . . .	A-12
A.2.1.2	SCALE (Scale) . . . . .	A-18
A.2.1.3	SHAPE (Shape) . . . . .	A-20
A.2.1.4	FFT . . . . .	A-29
A.2.1.5	UNSCRM (Unscramble) . . . . .	A-44
A.2.1.6	ORDER . . . . .	A-53
A.2.2	FIT Code . . . . .	A-70
A.2.2.1	MAGAPX (Magnitude Approximate) . . . . .	A-70
A.2.2.2	SMTGMX (Smooth and Tag Maximum) . . . . .	A-76
A.2.2.3	CKFMAX (Check FO Maximum) . . . . .	A-83
A.2.2.4	GMACPT (GAMMA Compute) . . . . .	A-92
A.2.2.5	FOLPL (FOLPL Compute) . . . . .	A-104
A.2.3	Subroutines . . . . .	A-111
A.2.3.1	SUBROT (Subroutine Block) . . . . .	A-111
A.2.3.2	DTASCL (Data Scale) . . . . .	A-124
A.2.3.3	SUBRT2 (Subroutine Block #2) . . . . .	A-138
A.2.4	Constants and Tables . . . . .	A-142
A.2.4.1	SYSSET (System Set) . . . . .	A-142
A.2.4.2	TRGTBL (Trig Table) . . . . .	A-146
A.2.4.3	LOGTBL (Logarithm Table) . . . . .	A-149
A.2.4.4	INSFIX (Instrument Fix) . . . . .	A-152

# APPENDIX B RUN TIME CODE

B.1	RUN TIME PROGRAMS . . . . .	B-1
B.1.1	SYSRUN (System Run) . . . . .	B-1
B.1.2	LDFIFO (Load FIFO) . . . . .	B-12
B.1.3	FFT COMPLETE TRACE . . . . .	B-22
B.1.4	FFT OUT TRACE . . . . .	B-30
B.1.5	FFT NO TRACE . . . . .	B-38
B.1.6	MULTI FFT . . . . .	B-41
B.1.7	8085 FULL TRACE . . . . .	B-44
B.2	RUN TIME TABLES . . . . .	B-46
B.2.1	SINWAV (Sine Wave) . . . . .	B-46

APPENDIX C RCA 1802 COSMAC Instruction Set

C.1 RCA 1802 COSMAC Instruction Set . . . . . C-1

APPENDIX D PROGRAM VALIDATION OUTPUT

D.1 SUBPROGRAM VALIDATION SERIES . . . . . D-2  
D.1.1 P-FFT, Impulse Input . . . . . D-2  
D.1.2 S-FFT, DC Input . . . . . D-5  
D.2 INPUT/OUTPUT VALIDATION SERIES . . . . . D-8  
D.2.1 P-FFT, Impulse to DC Series . . . . . D-8  
D.2.2 S-FFT, Impulse to DC Series . . . . . D-16

LIST OF FIGURES

1.3.a	15-channel ASP packaged for field use . . . . .	5
2.2.1.1.a	Original and Derived Time Series, with P and S-Wave Triggers and Detections . . .	8
2.2.2.a	RCA 1802 Development System Busses . . . . .	13
2.2.2.b	Seismic Data Processor Prototype System . . .	14
2.2.3.a	Initial ASP Prompt Message . . . . .	17
3.4.a	RCA 1802 CPU Architecture . . . . .	21
3.5.a	RCA COSMAC Development System . . . . .	24
4.2.a	System Program Maps (WORKER and BOSS) . . . . .	31
5.2.a	Examples of ASP Processing for two events . . .	72
5.2.4.1.a	GAMMA Calculation . . . . .	88
8.1.a	FFT Processing Times . . . . .	138





## 1.0 INTRODUCTION

### 1.1 Objective

The objective of this work was to develop a Fast Fourier Transform (FFT) and a Power Spectra Density Fitting program (FIT) for use in an Automated Seismic Processor (ASP) System for Microearthquake Networks. ASP is a multichannel system which digitizes and stores the readings from a number of distributed seismometers. Upon detection of an event of interest from a large enough number of sensors the FFT routine is called to calculate the Power Spectral Density of the earthquake energy for each of those sites. The resulting spectra is FIT for Long Period Level (LPL), Corner Frequency (F0), and High Frequency Slope (GAMMA), and these results are output to the operator.

The system is designed for long-term, remote, unattended sensing. The RCA 1802 CMOS microprocessor was selected, primarily because of its low power consumption. This processor, and the associated COSMAC architecture, are utilized for the FFT/FIT programs. Because the seismographic sensing capability must be deferred while the FFT/FIT programs are computing, the speed of the calculations is a critical parameter. Speed was optimized as much as possible within the additional constraint of a limited address space (due again to power consumption and also the limited availability of CMOS memory chips at the time of development).

## 1.2 Scope

This report will give a brief background of the requirement of the ASP development. A short introduction on the physics of earthquake dynamics is provided to set the context for operation. The interaction of the various sections of the ASP software system is presented. The operation of the FFT and FIT programs is then described in detail. Debugging tools which were developed for the task are explained. Results of the operation, both for tests of the FFT operation and for simulated earthquake operation, are presented. The Program Code, Run Time Code, FFT Validation Results and Earthquake Results are included.

## 1.3 Background

The background, purpose, design, and operation of the overall ASP System is beyond the scope of the work being reported on here. However, for the purpose of completeness, and to establish a point of reference for the reader, this section, and Sections 2.2 System Operation, 2.2.1 ASP Software, 2.2.2 ASP Hardware, and 2.2.3 ASP Operation, have been included. They are taken from [McEv] and are included with the permission of the authors.

For several years scientists at the University of California at Berkeley (UCB) and the associated Lawrence Berkeley Laboratory (LBL) have been involved in a research program to evaluate and develop seismological techniques for the exploration and delineation of geothermal reservoirs. It has been necessary in this effort to develop equipment as well as field procedures and theoretically based methods of analysis. From field investigations in Nevada [Maje78], at The Geysers [Maje79], and in the Cerro Prieto geothermal region [Maje80], it has become apparent that effective study of the relation between seismicity and

geothermal reservoir dynamics requires more than conventional data reduction techniques. The concept of an Automated Seismic Processor (ASP) was seen to offer the necessary rapid and cost-effective approach to microearthquake monitoring and analysis in applications to geothermal exploration and reservoir delineation.

The impetus for the development of ASP was the need to simplify routine microearthquake data acquisition and reduction in such investigations. Ideally, ASP would be a low-power, in-field data acquisition and real-time analysis system with adequate computational speed, bandwidth (0-100 Hz), and dynamic range (16-bit). The system should be multi-channel (up to 128), with the ability to operate unattended in hostile environments for long periods of time. Other automated systems are primarily laboratory-based systems compatible with larger computer facilities and peripheral devices (see [Stew] and [Alle]).

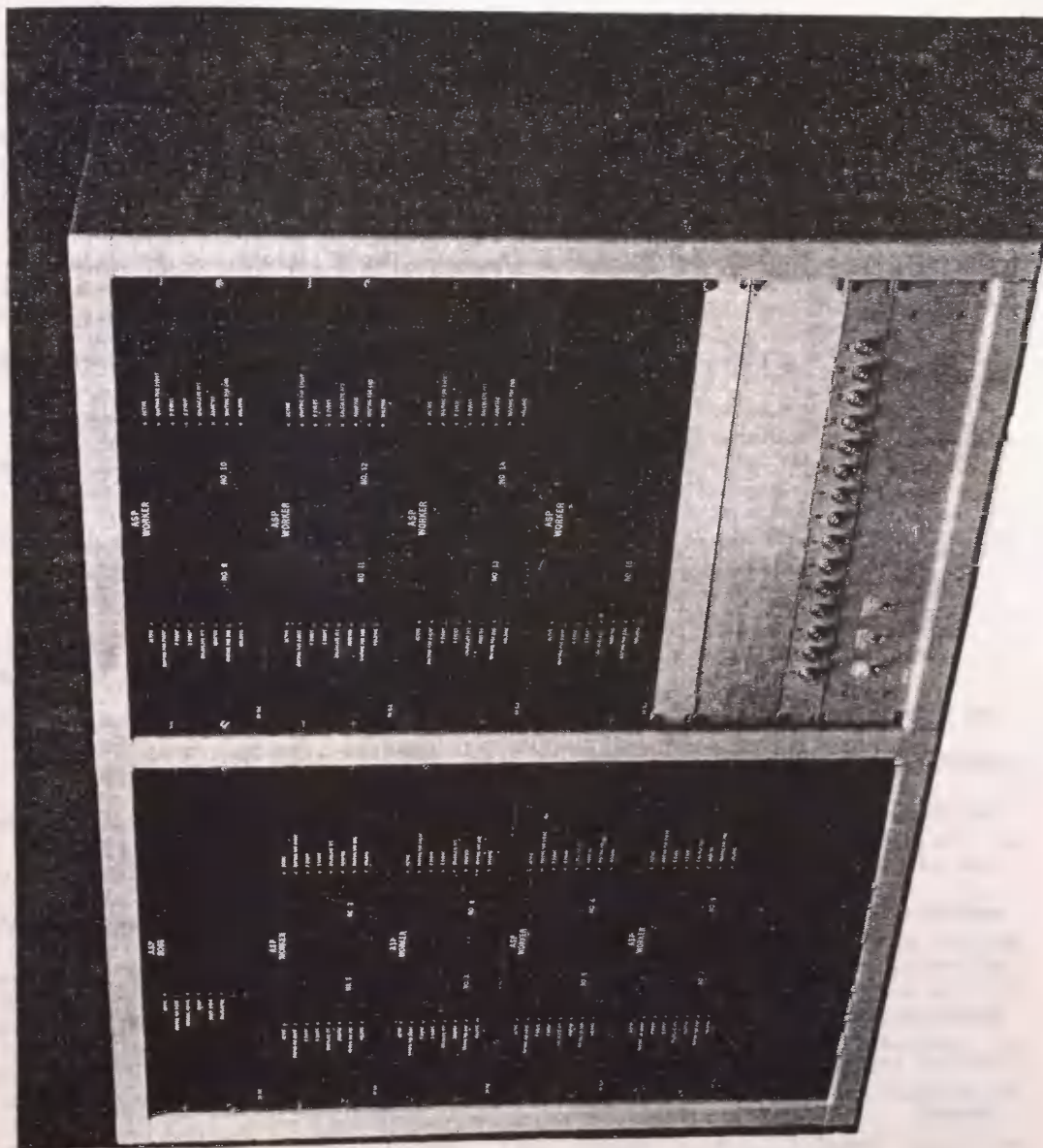
Our experience in microearthquake work has led to a conviction that it is possible to define and automate the associated routine time-consuming data reduction and processing tasks. For each data channel or station, these include the detection and identification of valid events, time and amplitude measurements of the P and S waves, calculation of the Fast Fourier Transform (FFT) on properly windowed P and S waves, and the reduction of P and S spectra to Long-Period Level, Corner Frequency, and High Frequency Slope. P-wave first motion polarity with a quality grade is also required. Given these data from a number of channels, the event would be located, with source parameters determined for the spectral information and the polarity data. An updated b value would be obtained for the sequence, along with statistics on the occurrence of events as a function of average S - P times.

## ASP FFT Program

### Introduction

Although other systems had demonstrated many of these capabilities, such real-time analysis had not yet been achieved with a low-power field-based system. Our most radical departure from previous work, however, is the philosophy that the original time series need not be saved. We are convinced that proper data reduction and judicious selection of output will provide virtually all results of interest in conventional microearthquake surveys. Based upon these concepts, ASP has been designed and built for routine microearthquake studies, with sufficient data analysis to provide critical earthquake parameters, in a trade-off with computing speed and capacity. (Figure 1.3.a shows a 15-channel ASP packaged for field use in a van).





## 2.0 ASP OPERATION

### 2.1 Earthquake Dynamics

Shock waves resulting from an earthquake can occur in two forms [Rich]. The initial wave is called a Primary or P-Wave and is characterized as compression in nature. There may also be a subsequent Secondary or S-Wave, which is characterized as shear. For these two waves the velocities of propagation are different (both are also dependent on the transmission strata). This difference in transmission times allows for a distance calculation and, if received by three or more stations, the hypocenter and/or epicenter may also be located. It will be seen that the difference in time of arrival is also used to select the correct window lengths for the FFT calculations.

### 2.2 System Operation

ASP was conceived as the low-power CMOS microprocessor technology revolution was just beginning. At the time the 8-bit CMOS RCA 1802 device was the only microprocessor that met our requirements. As a feasibility exercise, a two-channel prototype system using the 1802 was completed successfully in the fall of 1978. A 16-channel ASP was built the following year, in a joint hardware/software development program. While several circuit boards are RCA products, many are not "off the shelf", and were designed and built in-house. Approximately four man-years were required for the development, fabrication, and initial testing of the first 16-channel ASP. Hardware cost is about \$2500 per channel.



## 2.2.1 ASP Software

ASP is designed as a parallel processing device with a dedicated CMOS microcomputer monitoring each incoming channel of data. Each dedicated-channel microcomputer, or WORKER, feeds processed data to a central microcomputer, or BOSS, which controls the network and performs multi-station data analysis. A total of 127 WORKERS can be handled by a single BOSS (i.e., 128 individual processors).

### 2.2.1.1 WORKER Routines

The principal function each WORKER is to monitor the incoming time series of a data channel. The sequence of operations in the system is:

- (1) the time series  $X(t_i)$  is digitized at 100 samples/sec with 12 bit resolution;
- (2) the mean is removed from the time series;
- (3) a new time series  $X'(t_i)$  is formed by

$$X'(t_i) = (1/n) * \text{Sum}[X(t_j)], j = i-n, \dots, i$$

- (4) a Long Term Average (LTA) of 4096 points and a Short Term Average (STA) of 16 points is taken on  $X'(t_i)$ ;
- (5) if the STA exceeds the LTA by a specified constant, then a "trigger" is found, else WORKER waits for a new digitization point.

At any one time 512 points of the original demeaned time series plus the STA and LTA values are saved. If a trigger point is found the true P-Wave arrival time (PT) is computed based upon this trigger time and a different constant; it will always precede the trigger time. See Figure 2.2.1.1.a for two examples of original time series  $x(t_i)$ , derived time series  $x'(t_i)$ , trigger detection (second vertical line), and PT P-Wave arrival time (first vertical line).

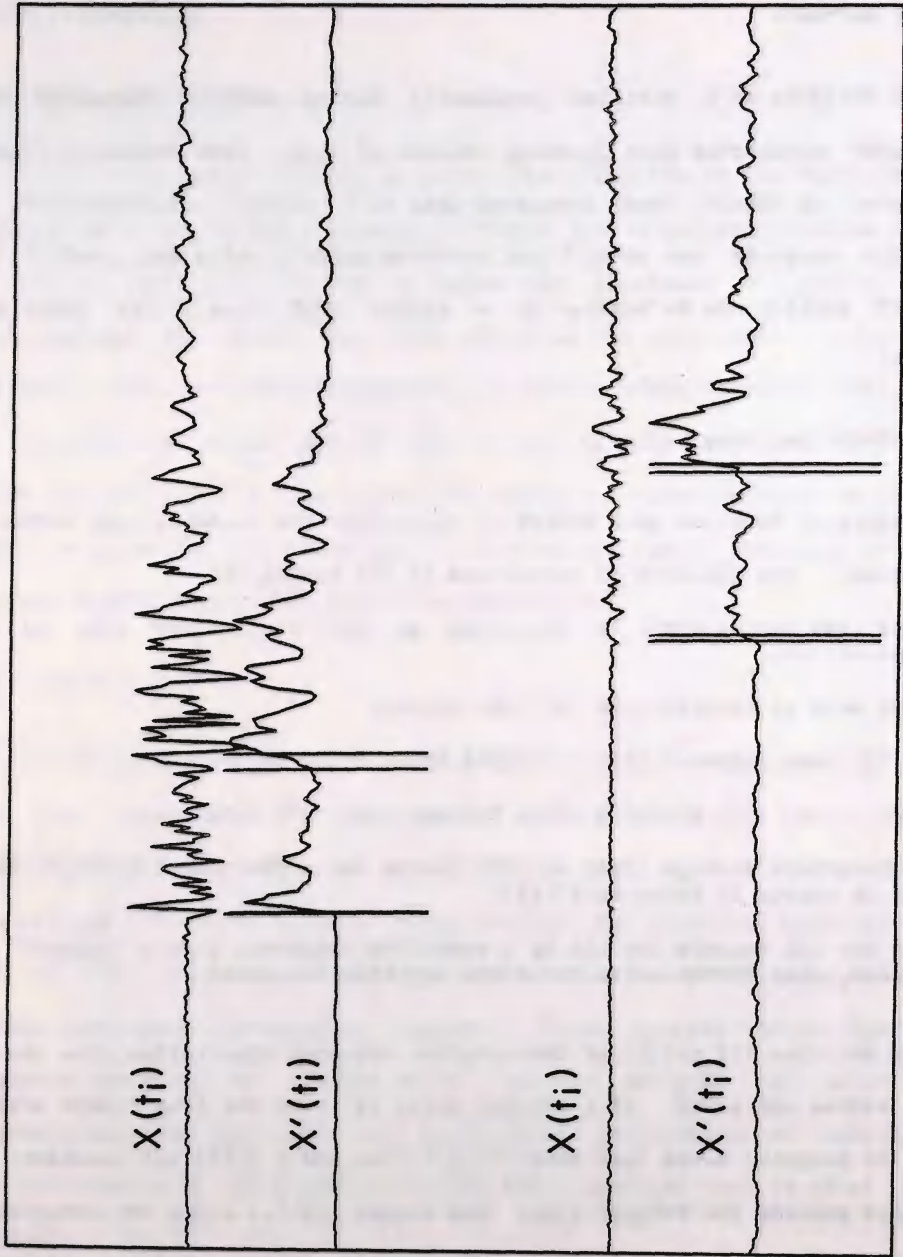


Figure 2.2.1.1.a Original  $x(t_i)$  and Derived  $x'(t_i)$  Time Series  
with P and S-Wave Trigger and Detection

XBL 817-3365

Although similar to the P-Wave algorithms, the S-Wave detection is slightly different. In this case the LTA is based upon the P-Amplitude (PA). When the STA of  $X'(t_i)$  exceeds PA by a specified amount then the S-Trigger is declared. S-Time of arrival, (ST), is based upon the slope of the pair of preceding four point sets. (Again see Figure 2.2.1.1.a for S-Wave trigger detections and ST arrival times, forth and third vertical lines, respectively). At this stage both P- and S-Waves are timed, DT, the ST-PT time, is computed, and the S-Amplitude (SA) is measured. Values of P- and S-Wave quality (PQ and SQ) and P-Wave polarity (PP) are also calculated.

Variable length windows for FFT computations are then placed around the P- and S-Waves. Available window lengths are 64, 128, and 256 points for the P-Wave and 128, 256, and 512 points for the S-Wave, selected automatically for ST-PT times of 0 to 1.28, 1.28 to 6 and greater than 6 seconds, respectively. If no S-Wave is found a 128 point window is used for the P-Wave. Demeaned data are held in these windows for subsequent spectral analysis.

FFTs are calculated for the windowed P- and S-Waves using the algorithm described by Brigham [Brig]. The first 12.5% of the S-Wave data window and the last 25% of both the P- and S-Wave data window are shaped with the raised cosine bell function  $[1/2 + 1/2(\cos(2\pi n/N))]$ . The amplitude response of a V-pole low pass Butterworth filter,

$$A = \frac{AO}{[1 + (f/FO)^{2V}]^{0.5}}$$

is fit for Long Period Level (LPL), Corner Frequency (FO), and the High Frequency Slope (GAMMA).

ASP FFT Program  
ASP Operation

A WORKER can be aborted at certain times in its calculation if the BOSS routine determines that too few stations recorded the event during a specified time window or that the time between the first and last P times was too small (i.e., a noise spike in the system). Upon abortion the WORKER returns to the detection mode.

An event is determined to be over when the signal level remains below the P-Trigger threshold for 256 samples. This can occur even without an S-Wave detection. If the level remains above the trigger threshold for six minutes a new LTA is calculated and the WORKER resets itself and resumes looking for another event.

All WORKERS will remain in HOLD while BOSS is calculating. While in HOLD the WORKERS can complete their sequence of calculations but cannot resume the detection mode until released by BOSS. This allows the BOSS to complete its calculations, output the results, and prepare to accept a new set of event data from the WORKERS. The total time for all WORKER operations for a typical microearthquake with ST-PT times less than 3 seconds is 10 to 20 seconds, depending upon FFT lengths.

In the system architecture each WORKER is capable of addressing 64K bytes of memory. Each worker as presently configured however contains only 12K of PROM and 4K of RAM, for a 16K memory size. Also, only 12-bit analog-to-digital converters were readily available in CMOS at the time of the design, although 16-bit arithmetic is used throughout the ASP. Integer arithmetic only, in twos-complement form, is used throughout WORKER software (with the exception of the GAMMA calculation), with division and multiplication in powers of 2 whenever possible. The FFT speed of 8 seconds for 512 points is achieved by using a



hardware multiplier board, designed and built in-house, with table look-ups for sine and cosine values.

#### 2.2.1.2 BOSS Routines

BOSS as the name implies, is responsible for overall system operation, event discrimination, and final calculation. To maintain overall system management on the message bus BOSS has the highest priority (priority 0), while each WORKER is assigned a specific priority (e.g., the number of each WORKER on the line) to insure smooth handling of messages.

Upon event detection each WORKER sends PT, PA, and PQ, which the BOSS stores. If enough "P-Messages" are received within a specified detection window the event is determined valid and P- and S-Messages (ST, SA and SQ) are recorded for another specified recording window. After this time BOSS begins calculating, depending on the modes of calculation the user has specified, selected from: (A) DEBUG; (B) EVENT COUNT; (C) B-VALUE; (D) LOCATION; (E) POLARITY; (F) FFT RAW DATA; (G) FFT STATION DATA; and (H) AVERAGE FFT DATA. Any or all of the modes can be selected and computed in any order so long as location parameters have been computed prior to their use in a subsequent mode. Because the RCA Floating Point Software has been replaced with a much faster Advanced Micro Devices 9511A Arithmetic Processor chip in the BOSS the processing time between events is less than 45 seconds when all 15 stations record an event. Output may be written to either a Silent 700 printer, a magnetic cassette, or both.

## 2.2.2 ASP Hardware

ASP utilizes the 8-bit CMOS RCA 1802 microprocessor, a byte-oriented central processing unit employing the COSMAC architecture. Double precision arithmetic is used throughout ASP to achieve 16-bit resolution. The overall system is shown in Figures 2.2.2.a and 2.2.2.b.

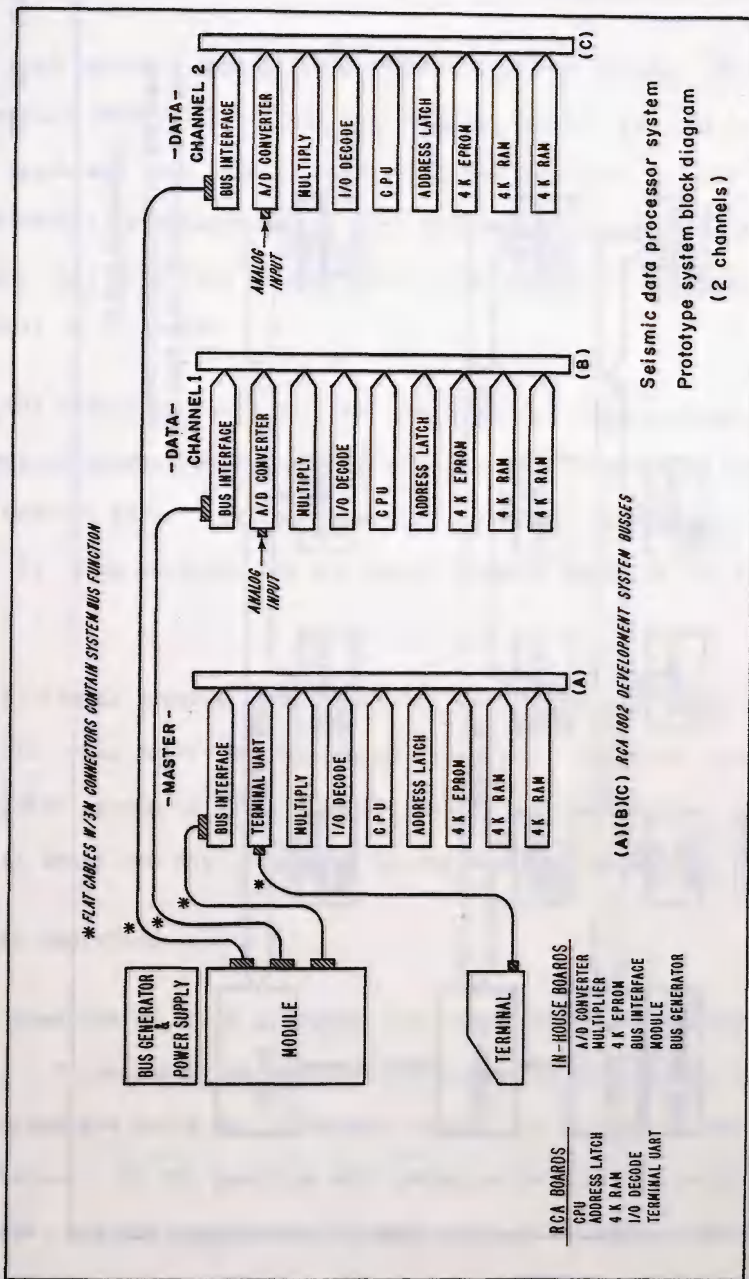
Each WORKER uses 10 different types of circuit boards, 6 designed and fabricated in-house, the rest are RCA products. The boards are: (1) CPU Board (RCA); (2) Address Latch and Bank Select (RCA); (3) I/O Decode (RCA); (4) 4K CMOS RAM (RCA); (5) Analog-to-Digital Converter; (6) Bus Interface 1; (7) Bus Interface 2; (8) PROM board; (9) 16 x 16 Multiplier; and (10) the Status Display board. With 3 PROM boards in each WORKER (12K), there are a total of 12 boards per WORKER.

The Analog-to-Digital Converter board uses a 12-bit CMOS Datel unit (ADC-HC12B) with a Datel sample-and-hold unit (SHM-LM-2). It operates at 100 samples/sec with a full scale input of +/- 5 volts, for a 2.5 millivolt least significant bit.

The Multiplier board is the only non-CMOS board in WORKER. Based on the Advanced Micro Devices 25LS14 with a 25LS22 shift register, it performs a 16 x 16 bit integer multiply in 30 microseconds, and is powered up on demand.

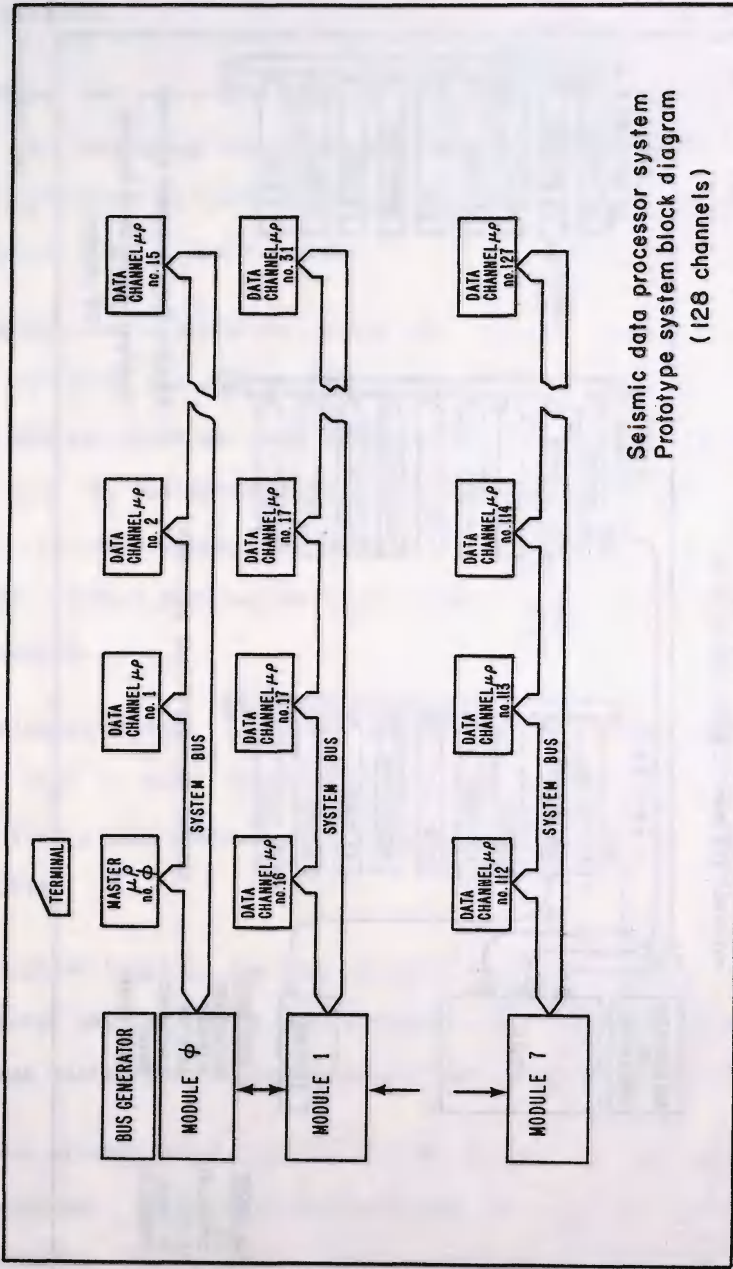
The Status Display board controls a bank of LEDs to indicate the program position in WORKER. The various status lights are: ACTIVE; WAITING FOR EVENT; P-EVENT; S-EVENT; CALCULATE FFT; ABORT; WAITING FOR END; and HOLD.





XBL 7812-14137

Figure 2.2.2.a RCA 1802 Development System Busses



XBL 7812-14138

Figure 2.2.2.b Seismic Data Processor Prototype System

The BOSS hardware employs 12 different types of boards. They are: (1) two UART boards (RCA); (2) two RAM boards (RCA); (3) I/O Decode (RCA); (4) Address Latch and Bank Select (RCA); (5) Bus Interface 1; (6) Bus Interface 2; (7) Arithmetic Processor Board; (8) Real Time Clock Interface; (9) Time-Code Interrupt, (10) four PROM boards; (11) Status Display; and (12) the CPU board, for a total of 17 boards.

The Bus Generator board provides the 2 MHz basic clock signal to the system. It is crystal-controlled and provides four control signals to the system at a 10 KHz bus message rate: latch request; source; destination; and transfer pulse. It also provides the ADC sample command pulse to the system at 100 per second.

The system is powered by a 25 watt 12 volt to +5 volt DC-DC converter and a 6 watt 12 volt to +/- 12 volt DC-DC converter. The power requirement of each microcomputer system is about 1 watt. One 12 volt automobile battery can run ASP for at least one day, depending on the activity level.

### 2.2.3 ASP Operation

Upon power-on or reset a prompt for user input is printed (see Figure 2.2.3.a). In response the operator may choose to do nothing, which will result in all parameters being set to default values, or he may set the parameters to desired values. An "S" response will prompt a request for sampling rate, number of stations, and the coordinates of each station, as well as a time correction value for each station. Level 1, Level 2, and WORKER parameters may also be set (relating to time values, velocity components, quality factors, trigger levels,

ASP FFT Program  
ASP Operation

etc.). Modes may also be selected (see BOSS Routines Section 2.2.1.2 above). Settings may be changed at any time. As can be seen ASP offers flexibility in field procedures by providing the user with results during the experiment, in a real-time mode.

AFTER \* ,TYPE:

S TO SET STATION PARAMETERS  
W TO SET WORKER PARAMETERS  
1 TO SET LEVEL ONE PARAMETERS  
2 TO SET LEVEL TWO PARAMETERS  
P TO CONTROL PRINTOUT  
M TO SET MODE  
H TO PRINT THE ABOVE

!! ENTER ALL NUMBERS AS FOLLOWS !!

SMMMMMSEE S=SIGN, + OR -

M=MANTISSA, 5 DIGITS

E=EXPONENT, 2 DIGITS

DECIMAL POINT IS S.MMMMMSEE

FOLLOW EACH NUMBER WITH RETURN

\* S

ENTER STATION PARAMETERS:

(SETS ALL OTHER PARAMETERS TO DEFAULT)

SAMPLES/SECOND ? +10000+03

NUMBER OF STATIONS ? +40000+01

FOR EACH STATION, ENTER COORDINATES,

ONE/LINE IN THE ORDER:

SX, SY, SZ, SC

STATION +10000+01

+33500+01

+60000+01

+00000+00

+00000+00

DATA OK (Y/N/S S-STOPS) ? Y

Figure 2.2.3.a Initial ASP Prompt Message



### 3.0 FFT/FIT PROGRAM DESIGN

#### 3.1 WORKER Operation

The WORKER operation prior to the FFT/FIT call been described previously. To summarize - the demeaned data is written to the YP-FIFO. Upon the detection of a P-Wave the first word address of the P-Wave data is determined. The demeaned data continues to be written awaiting an S-Wave. If an S-Wave occurs the data is written into the YS-FIFO, the first data address of the S-Wave is determined, and the time difference of arrival (DT) are calculated. Then the FFT/FIT is called, first for the S-FFT (if it occurred), then for the P-FFT.

#### 3.2 FFT/FIT Requirements

Again, some of the FFT Requirements have been mentioned already. The primary requirement is for speed since while the FFT/FIT is calculating the sensor system is off-line. Therefore coding organization and efficiency are at a premium. As will be seen (RCA 1802 Architecture Considerations Section 3.2) to speed operations no subroutine calls are used but control is passed through a register transfer. For speed (and mathematical accuracy) table look-ups are used for the trigonometric, logarithmic, and instrument correction functions.

The FFT was designed as an In-Place Decimation-In-Frequency function. Real data was treated as Real and Imaginary pairs and unscrambled at the end. Both decisions (in-place and Real as Real/Imaginary) were taken to lessen the memory requirements.



The output spectrum will be FIT for Long Period Level (LPL), Corner Frequency (FO), and the High Frequency Slope (GAMMA). The FIT programs check for the validity of the results, and return with an error flag if out of range.

### 3.3 Mathematical Precision

Consideration must be given as to the impact of the finite register length representation of the coefficients and (more importantly) the data in the iterative FFT algorithm. (For a mathematical treatment of this topic see [Kane], [Oppe72], [Oppe75], [Rabi], [Wein69a], [Wein69b], and [Welc]). The primary source for error in the FFT algorithm is the truncation necessary to assure no overflow errors in the multiply and accumulate operation of the butterfly calculation.

The method utilized to achieve the maximum dynamic range in the numbers was to scale the data as often as possible. Prior to any calculation which operates on the entire FIFO the data is checked, and scaled if possible, to the maximum the coming operation allows while assuring there will be no overflows. This scaling quantity is carried through the FFT/FIT operation (TOTLSH, Total Shift). This is in effect a Block Floating Point numerical representation.

Analysis of the resultant error has been calculated by Welch [Welc]. Theoretical Upper and Lower Bounds have been calculated and Experimental results for various distributions of input data have been developed. Listed below is the RMS(error) to RMS(result) ratios for the Upper Bound and the Experimental Output (for a random number input with zero mean and uniform distribution over  $(-1,1)$ ) for 16-bit quantization.

N	RMS(error)/RMS(result) x 10 <sup>4</sup>	
	Upper Bound	Experimental Result
64	1.8	0.8
128	2.4	1.1
256	3.6	1.2
512	5.2	1.4

This error must be viewed in the context of the FIT operations to follow. As will be seen the first FIT function approximates the magnitude of the frequency components by a very rough function (see MAGAPX, Section 5.2.1). Then the Long Period Level (LPL), Corner Frequency (F0), and the High Frequency Slope (GAMMA) will be approximated, this time on averaged data. The geophysicists on the project, based upon work done on a large mainframe, are comfortable that these approximations are not enough to cause significant error in the results. Based upon this opinion the truncation error was also determined to be not a significant factor.

### 3.4 RCA 1802 Architecture Considerations

The RCA 1802 COSMAC Architecture and instruction set is quite different from the more traditional microprocessors. Its primary benefit, indeed the reason why it was chosen, is its low power consumption. The CPU has the internal architecture shown in Figure 3.4.a. It consists of:

- an 8-bit Accumulator (D),
- a 1-bit Carry Bit (DF),
- an Interrupt Enable Flipflop (IE),
- sixteen 16-bit Register Files (R0, ..., RF) any of which may serve as Data Registers, Pointers to Memory, or as the Program Counter,
- a 4-bit pointer (P) to the Register File to select the Program Counter,

- another 4-bit pointer (X) which selects the register to be used as a pointer to an operand in memory,
- an 8-bit Temporary Storage register (T) used to hold X and P when an Interrupt occurs,
- four internal Flags (EF1, ..., EF4) which are external pins on the CPU chip and may be tested externally, and
- an internal Flag (Q) which may be set, cleared, and tested under program control and which also drives an output line from the CPU chip.

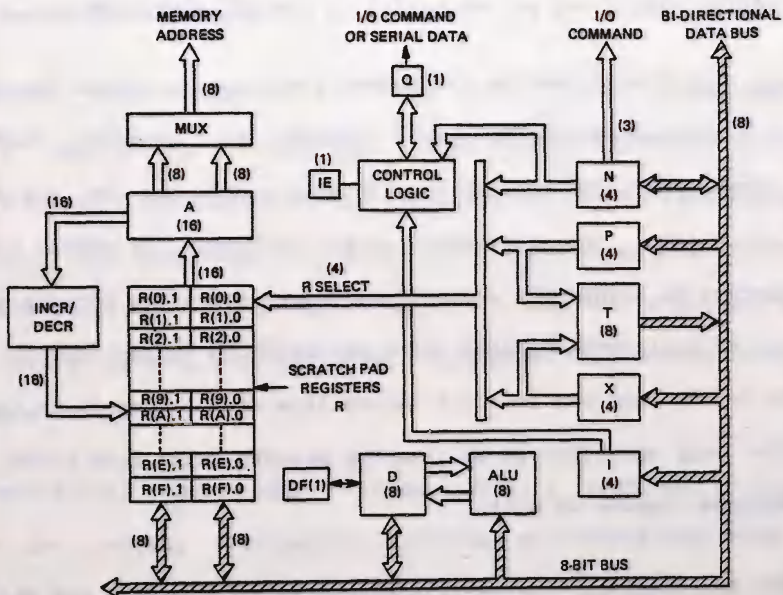


Figure 3.4.a RCA 1802 CPU Architecture

The Instruction Set is listed in Appendix C. The Instruction Set is relatively efficient. There are only seven 3-byte instructions, while over half of the instructions are 1-byte. However the set is less powerful than other processors.

The 16-bit register associated with some instructions is indicated by the generic label RN, which may be any of the registers R0 to RF. Some instructions use a selected register as a pointer to memory. Thus M(RN) indicates the memory location pointed to by the register RN. In addition M(RX) indicates the memory location pointed to by the register which is pointed to by the Register Pointer X. This indexed addressing proved useful in the FFT algorithm.

The most significant benefit the COSMAC architecture offers is the ability to change the program counter by simply changing the P pointer. This allows for very fast subroutine calls because the CPU status need not be saved for this transfer of control. The programmer has the obligation of course to keep track of the registers to assure the subroutines do not overwrite necessary data. It is this type of fast, tight coding which was required on the ASP FFT program and indeed this is the only way in which subroutines are accessed in the program body. (The only exception to this is the primary calls from FFTMN to the main program sections, twelve in all).

At least two difficulties were evident in the Instruction Set with regard to the FFT algorithm. The first, which would apply to any program utilizing the COSMAC, is the lack of orthogonality or regularity in the instruction set. For example, there are four Load D operations: Load M(RN); Load M(RN) and Increment RN; Load M(RX); and Load M(RX) and Decrement RX. However there are only two Store D operations: Store to M(RN) and Store to M(RX) and Decrement



RX. Such lack of orthogonality or regularity makes efficient programming difficult.

The second difficulty in the Instruction Set is the lack of relative addressing, that is pointing to memory by an increment from some base address. That operation occurs continually in any Signal Processing application such as the FFT (for example the calculation of the butterfly pair's addresses). The requirement to continually calculate the exact address, then access it as  $M(RN)$ , led to a significantly greater amount of coding.

### 3.5 Development System Support

Development System support was very poor. The Development System (CDP18S005) contained a 18S102 CPU, 64K of memory, and two 256K floppy disks (see Figure 3.5.a) downloadable to the Prototype ASP System (see Figures 2.2.2.a and 2.2.2.b). File management in memory and on the floppies is nonexistent. In all cases the programmer had to direct where the data was to go, and do it right or else. Careful back-up procedures were essential. A line editor was the only one available.

More importantly there was no debug support provided. There was no capability for setting breakpoints, testing or setting registers and memory locations, or for restarting the program. Run Time code had to be developed and interfaced with the program with the use of manually inserted Long Branches (see Run Time Programs Section 6.0). These Long Branches were originally overwritten onto the code; later No-Operations (NOPs, #C4C4C4 for the 1802) were inserted at strategic locations for the breakpoints. In this way the FFT/FIT subprograms were cut up into manageable (debuggable) sections. These techniques later

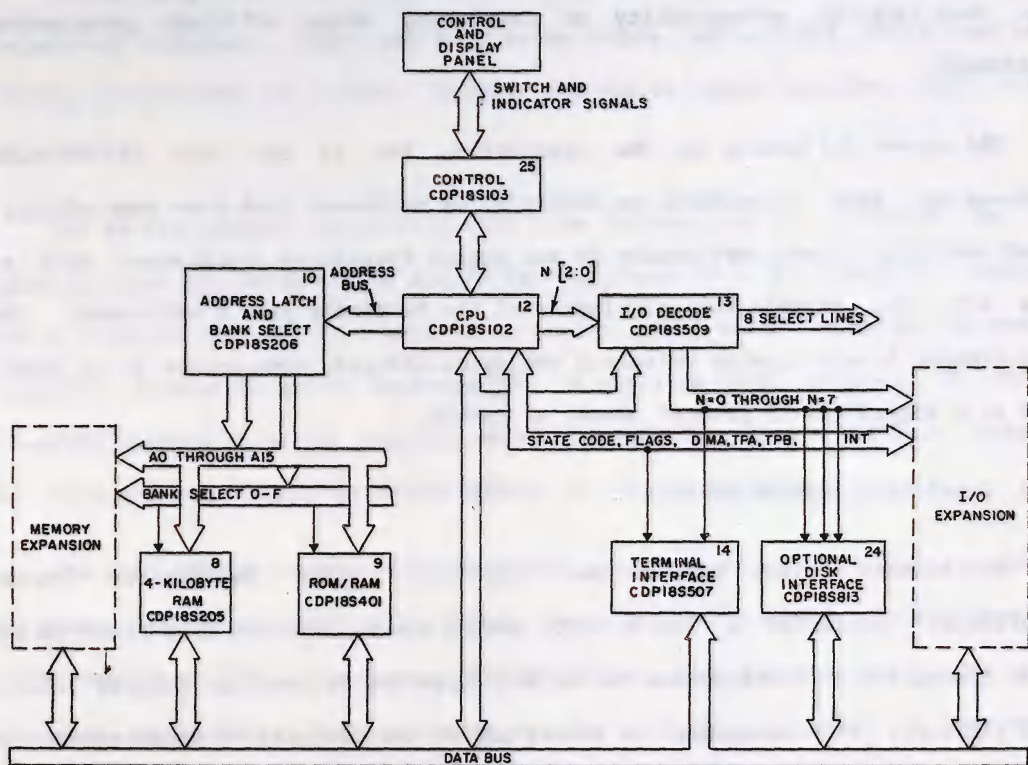


Figure 3.5.a RCA COSMAC Development System

proved essential in the trapping of an intermittent error regarding FIFO wraparound (see Section 6.6).



### 3.6 Programming Techniques

#### 3.6.1 Structured Programming

Because of the recognition in recent years of the cost of maintenance of computer software, techniques have been developed to make programs more intelligible and modifiable. Two of the most important concepts are those of Structured Programming and Data Hiding. The basic idea of Structured Programming is the use of a Top-Down Design approach where a large program is divided hierarchically into smaller and smaller sections until a "primitive" level is reached. These primitives have a well defined entry point, a limited number of global and local variables to be manipulated, and a singular exit point. This is coupled with and elegant Control Structures involving "if...then...else", "do...while", "do...until", "case a, b, ...else", and other constructs. (A result of this last statement is the lack of "GOTOs" in the program). The code should also be self-documenting.

Another consideration for flexible, maintainable code is the separation, as much as possible, of the program algorithm structure from data structure (Data Hiding). Data should only be manipulated at the lowest possible primitive level. This allows for later changes in either domain (algorithm or data) with less of an impact on the other.

Most of these rules are broken in the FFT/FIT program. Top Down design, with modular primitives and self-documenting code, was used once it was appreciated (see Human Factors Section 3.7). But the control structures were not used - GOTOs were. Also the program structure was very closely tied to the data structure, indeed inexorably tied to it.

There were some good reasons for the use of this programming style. The FFT/FIT program was developed to be fast, tight code as opposed to the Structured Programming elegant, maintainable style. The heart of the FFT is not expected to be changed in the foreseeable future (yes ... they all say that). Linking with the data structures was necessary to get the job done, and essential to give it any speed. Finally, there is no support for the control structures within the 1802 Assembly Language (and it would have been too slow to program it in).

### 3.6.2 Numbering Notations

Some Numbering Notations were used in the program which warrant explanation. Some of them were supported by the Development System and are in the code; some are adopted and found only in the program comments.

Numbers are represented as decimal unless preceded by the "#" sign, in which case they are hexadecimal. All numbers are Twos-Complement Fixed Point (usually 16-bit), with the exception of some numbers in the FIT routines. For the Long Period Level a 24-bit floating point construct was adopted for the Integer and Exponent. For the High Frequency Slope and the Logarithmic functions there was an Integer/Fraction scheme.

Variables are limited to six characters. Upper case characters only were allowed. (Both of these were more of a limitation to clear, understandable code than was expected and another violation of Structured Programming rules). To aid in clarity the acronym is usually defined in the program header or in the code close to its point of initial use. Program labels are any unique six character acronym followed by a colon (":").

Notations for Addresses are defined in the Development System as ,A(ADDR) for a full word address and A.1(ADDR) and A.0(ADDR) for the high and low bytes respectively. Similar schemes were used for Registers, R(NAME) being the data word in a register and R.1(NAME) and R.0(NAME) being the high and low bytes. Similarly DATA.1 and DATA.0 represent the high and low bytes of data. M(ADDR) is the value at that memory address; this data word is often used as a pointer value. Therefore M(M(ADDR)) and M(R(NAME)) is the value being pointed to by the memory pointer or the register pointer.

### 3.6.3 Programming Conventions

Most of the Programming Conventions have been mentioned elsewhere; they are listed here for reader convenience. The first is the use of a common program header, listing the Program Constants and Global Variables. Versions of this header were maintained from one major program revision to the next, then modified for all sections of code.

Registers FRP and ZAP1 were addressed to the Fast RAM section #30xx, the location of most of the variables. This allowed for addressing the variables by pointing the lower register's lower byte only, thus saving instructions and time.

As one concession to convenience Long Branch instructions were used instead of Short Branches in the subprograms. Only one more instruction byte is used and the speed lost was not dramatic. This allowed the relocation of code without concern for page boundaries. In the subroutines however, because they were called so often, Short Branches were used and care was taken to avoid page boundaries.



The traditional Call/Return sequence was not used for subroutine accesses. Instead the Program Counter Pointer P was directed to another register (usually ZAP, thus SEP ZAP) which contained the subroutine address. Upon return ZAP was usually left readdressed for another subroutine access.

Interrupts by the WORKER for new data values were allowed during FFT/FIT computation to maintain the Long Term and Short Term Averages. Interrupts were disabled during multiplications however as the hardware board required the use of the I/O channels; interrupts would have invalidated the results.

During Software Development patches to the code were invaluable for programming convenience. These were implemented in the form of Overlay Programs which were loaded into memory after the program code. They consisted of Long Branches to an unused section of memory, with the modified code resident there, followed by Long Branch back to the main code. (See FFTMN Section 5.0.1 for an example of three patches).

### 3.7 Human Factors

Comment should be made concerning the programming education which occurred on this project. This was the author's first program of any consequence (he would term this program large for a single programmer) and his first assembly language program. It was developed without much assistance and could be likened to learning how to swim by being thrown into the deep end of the pool. You finally learn how but not after much splashing about and swallowing of a lot of water. Then after some longer period of time you finally become kind of elegant. It is left to the discretion of the reader whether elegance was approached.

Along this line is an observation that although Top Down Design is obviously the correct way to develop a program it is only possible if one is capable of building from the Bottom Up once the lowest level is reached. That is you can't do it until you know how to do it, and you can't know how to do it until you do it. Stated another way the original sections of code do not demonstrate Top Down structure, if they represent any structure at all. Only with some experience can one program (or do anything) with confidence. But then Master's work is meant to educate, isn't it. This was a good education.



## 4.0 DATA STRUCTURES

### 4.1 Register Names

Valu	Name	Description
-----	-----	-----
0000	ZAP	RO is the Scratch Pad Register
0003	PC	Normal Program Counter
0007	DSP	Data Stack Pointer
0008	FRP	Fast RAM Pointer
0009	MP	Message Pointer
000A	ZAP2	Arithmetic Scratch Pad Register
000B	ZAP3	Arithmetic Scratch Pad Register
000C	ZAP1	Also used as Fast RAM Pointer
000D	MA	Arithmetic Memory Address Pointer
000E	MQ	Arithmetic Accumulator Extension
000F	AC	Arithmetic Accumulator

### 4.2 Memory Allocations

The WORKER has 12K of ROM, addressed from #0000 to #2FFF, and 4K of RAM, addressed from #3000 to #3FFF. See Figure 4.2.a for the System Program Maps for the WORKER and the BOSS. The Development System has 64K of memory, thus Development Programs and Tables can use addresses above #4000 (see the Run Time Programs Section 6.0).

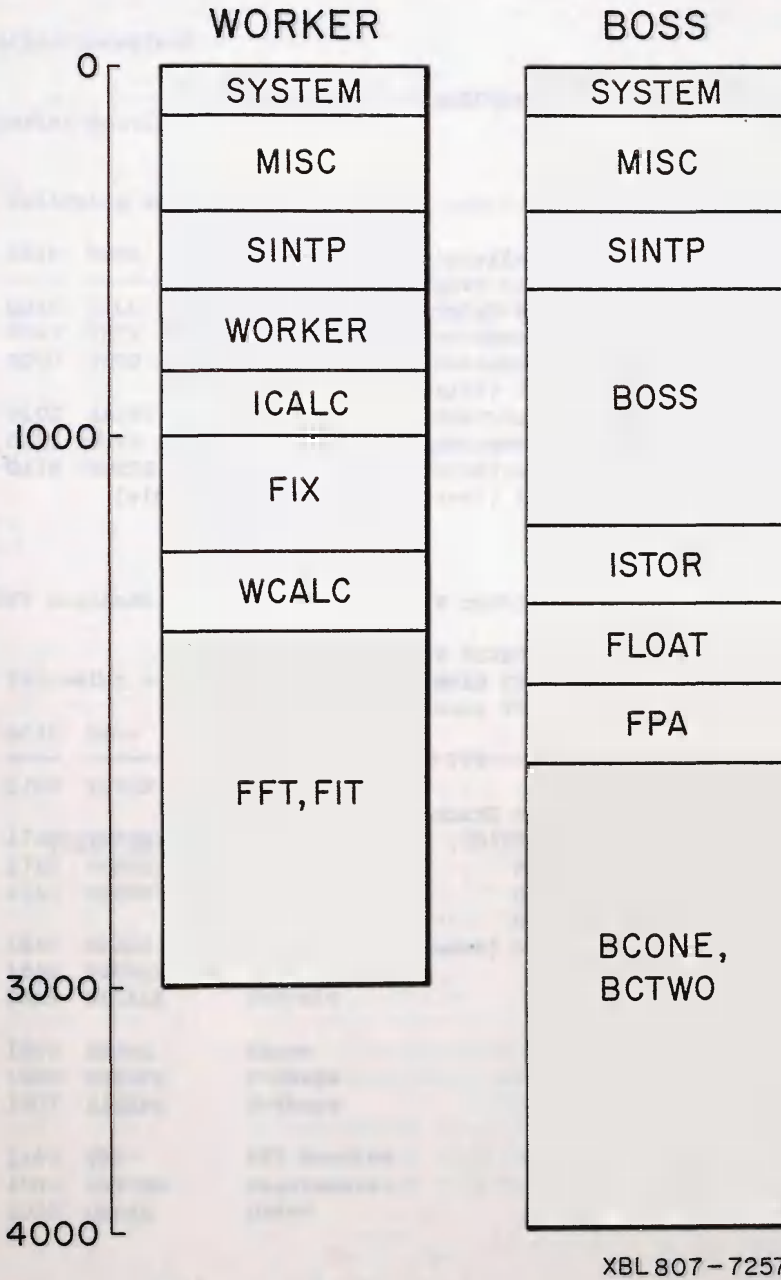


Figure 4.2.a System Program Maps (WORKER and BOSS)

# WORKER ADDRESS MAPPING

0000 - 3FFF      Memory

0000 - 2FFF      ROM

0000 - 0031	Initialization Program
0032 - 030F	Utility Programs
0310 - 16FF	WORKER Programs
1700 - 2637	FFT Programs (Set #1)
2660 - 26F6	FFT Constants (SYSSET)
26FE - 27FF	TRGTBL (Trig Table)
2800 - 2BC1	FFT Subroutines
2C00 - 2CD8	FFT Programs (Set #2)
2D00 - 2DFF	LOGTBL (Logarithm Table)
2E00 - 2FFF	INSFIX (Instrument Correction Table)

3000 - 3FFF      RAM

3000 - 30FF      FRAM (Fast RAM)

3000 - 30C3	WORKER Variables
30C4 - 30DF	FFT Global Variables
30E0 - 30FF	FFT Local Variables

30E0 - 30FA      FIT Global Variables

3101 - 3370	WORKER Stacks and Buffers (STACK, DSTACK, BUSS, UART, and MISC)
3371 - 3600	Unused
3601 - 3A00	YPFIFO
3A01 - 3E00	YSFIFO
3E00 - 3FFF	Unused (some DTASCL Variables)

### 4.3 Routine Locations

#### 4.3.1 Worker Routines

The following WORKER routines are of use to the FFT/FIT programs.

Addr	Name	Description
0033	CALL	Address of Call Routine
0043	RETN	Address of Return Routine
0050	INTP	Address of Interrupt Routine
010C	XBUSS	Address of Buss Transmit Routine
0115	MOVE	Address of Block Move Routine
0118	BMESS	Address of Buss Message Form Routine

#### 4.3.2 FFT Routines

The following are the FFT Routines, with their memory locations.

Addr	Name	Description
1700	FFTMN	FFT Main
17A0	LNSET	Length Set
17A0	PLNSET	P-Length Set
17A3	SLNSET	S-Length Set
18A0	SCALE	Scale
18A0	PSCALE	P-Scale
18A3	SSCALE	S-Scale
1900	SHAPE	Shape
1900	PSHAPE	P-Shape
1903	SSHAPE	S-Shape
1A60	FFT	FFT Routine
1C00	UNSCRM	Unscramble
1D20	ORDER	Order

#### 4.3.3 FIT Routines

The following are the FIT Routines, with their memory locations.

Addr	Name	Description
2000	MAGAPX	Magnitude Approximate
2100	SMTGMX	Smooth and Tag Maximum
2250	CKFMAX	Check FO Maximum
23FO	GMACPT	GAMMA Compute
2C00	FOLPL	FOLPL Compute

#### 4.3.4 Run Time Routines

The following programs were used during program development and may be loaded into the Development System memory at locations above the WORKER locations (above #4000).

##### System Run Functions.

Addr	Name	Description
7000	SYSRUN	System Run (to #7400)
7003	RGPRNT	Register Print, then Quit
7006	PRGDUN	Program Done, so Quit
7009	RGDATA	Print Register Data
700C	FFTRAM	Print FFT RAM
700F	PFTDTA	Print P-FFT Data
7012	SFTDTA	Print S-FFT Data
7015	FFTDTA	Print FFT Data

##### Input Data Forcing Functions.

Addr	Name	Description
5100	LDFIFO	Load FIFO (to #5244)
5100	YPZERO	Zero YPFIFO
5103	YSZERO	Zero YSFIFO
5106	MANINP	Manual Input
5109	MODSIN	Modified Sine Input



## 4.4 Stacks, Buffers and FIFOs

### 4.4.1 Stacks

The following WORKER stacks are not used by the FFT/FIT routines but are listed in the program headers, and are included here for completeness.

Addr	Name	Description
----	-----	-----
3140	STACK	Stack goes downward for 64 bytes
3180	DSTACK	Data Stack goes downward for 64 bytes

### 4.4.2 Buffers

The following WORKER Buffers are listed in the FFT/FIT program headers. They are not directly used by the FFT/FIT routines (although the Buss Transmit buffer is used by the BMESS WORKER routine, which is called by FFTMN to transmit the FFT/FIT results to the BOSS).

Addr	Name	Description
----	-----	-----
3181	BBUFF	Begin Buffer Space
3181	XFWA	Buss Transmit Buffer First Word Address
3200	XLWA	Buss Transmit Buffer Last Word Address
3201	RFWA	Buss Receive Buffer First Word Address
3280	RLWA	Buss Receive Buffer Last Word Address
3281	IFWA	UART Input Buffer First Word Address
32D0	ILWA	UART Input Buffer Last Word Address
32D1	OFWA	UART Output Buffer First Word Address
3320	OLWA	UART Output Buffer Last Word Address
3321	SFWA	String Buffer First Word Address
3370	SLWA	String Buffer Last Word Address

#### 4.4.3 FIFOs

FIFOs are used for the recording of the (demeaned) seismometer readings. As described in Section 2.2.1.1 the data is written into the YPFIFO until a trigger is detected. Data continues to be written there until either an S-Wave is detected or until the event is over (no S-Wave). If an S-Wave is detected the data is then written into the YSFIFO. The FFT/FIT program is then called, first for the S-FFT (if it exists) and then for the P-FFT.

FIFOs are defined by nine byte tables as follows:

bytes 1,2    Address of Oldest Data (H,L)  
bytes 3,4    FIFO LWA+1 (H,L)  
bytes 5,6    FIFO FWA (H,L)  
bytes 7 8,9   FIFO Running Sum (L,M,H)

Addr	Name	Description
30A5	YPTBL	YPFIFO Table
30AE	YSTBL	YSFIFO Table

The FIFOs themselves are at...

Addr	Name	Description
3601	YPFWA	YPFIFO First Word Address
3A00	YPLWA	YPFIFO Last Word Address (YPFWA + 1023)
3A01	YSFWA	YSFIFO First Word Address
3E00	YSLWA	YSFIFO Last Word Address (YSFWA + 1023)

## 4.5 Variables

### 4.5.1 WORKER Variables

WORKER Variables used by the FFT/FIT programs are contained in FRAM (Fast RAM). They are defined at the beginning of the common FFT/FIT program header, as shown in the listings in Appendix A.

Addr	Name	Description
-----	-----	-----
3046	SLAST	Time for the Last YSFIFO
3048	DT	ST Minus PT

### 4.5.2 Global Variables

#### 4.5.2.1 FFT/FIT Global Variables

Fifteen Global Variables are defined in the System Header. They are all contained in FRAM and are accessed by many of the FFT and FIT programs.

#### Flags

Addr	Name	Description
-----	-----	-----
30C4	PFTFLG	P-FFT or S-FFT Flag
30C5	BDRYOK	Boundary OK Flag

#### Move Variables

Addr	Name	Description
-----	-----	-----
30C6	FREELN	Free Length Count
30C8	FWA	First Word Address of Move Out FIFO
30CA	MOVECT	Move Count

#### Length Variables

Addr	Name	Description
-----	-----	-----
30CC	PFFTLN	P-FFT Length
30CE	SFFTLN	S-FFT Length
30D0	FFTLN	FFT Length

ASP FFT Program  
Data Structures

Length Pointers

Addr	Name	Description
30D2	PLNPTR	P-FFT Length Pointer
30D4	SLNPTR	S-FFT Length Pointer
30D6	FLNPTR	FFT Length Pointer

Start Address for FFT Data

Addr	Name	Description
30D8	BASE	Start Address for FFT Data

Scale Values

Addr	Name	Description
30DA	TOTLSH	Total Shift Value
30DC	PSHIFT	P-FIFO Shift Value

SHAPE/FFT Value

Addr	Name	Description
30DE	CSVLU	Cosine Value Temporary Storage

#### 4.5.2.2 FIT Global Variables

The following FIT Global Variables are actually maintained in the FIT programs as Internal Variables, but since they are in all the program sections they should more correctly be considered Global.

The following values are intermediate results for FIT calculations.

Addr	Name	Description
----	-----	-----
30E0	MXADDR	Address of Maximum
30E2	FMAX	Frequency Index of Maximum
30E4	MXVALU	Maximum Value (Integer)
30E6	MXEXP	Maximum Value (Exponent "Shift" Value)

Values being calculated for transmittal to BOSS.

Addr	Name	Description
----	-----	-----
30E8	LNFFT	Length of FFT
30EA	F0	F0, Corner Frequency
30EC	GMAINT	GAMMA Integer
30EE	GMAFRC	GAMMA Fraction
30F0	MXNINT	Maximum Energy Integer
30F2	MXNEXP	Maximum Energy Exponent

#### 4.5.3 Local Variables

Local Variables, called Internal Variables in the Program sections, reside in addresses #30E0 to #30FF. They are listed in those sections describing the individual Programs.



## 4.6 Constants

### 4.6.1 FFT/FIT Constants

A number of FFT/FIT constants are used for programming efficiency and flexibility. They involve such values as addresses, pointer values, skip counts, etc.. They are defined in the Program Header, as seen in any of the programs included in Appendix A. The values are set by the SYSSET (System Set) program, also in Appendix A. They occupy addresses #2660 to #26F5 in ROM.

Addr	Name	Description
-----	-----	-----
2660	FFTTBL	FFT Tables (#2660 to #27FF)

#### Length Set Tables

The Length Set Tables are defined by ten byte directories as follows:

bytes 0,1 DT Length Boundary  
bytes 2,3 P-Length Value  
bytes 4,5 S-Length Value  
bytes 6,7 Pointer into P-Shape Table  
bytes 8,9 Pointer into S-Shape Table

Addr	Name	Description
-----	-----	-----
2660	PDEFLT	Solo P-Wave Default Length
266A	DT128	DT $\leq$ 128
2674	DT600	128 < DT $\leq$ 600
267E	DTMAX	600 < DT $\leq$ 32K

#### P and S Shape Tables

First the P-Shape Function

bytes 0,1 Count for the 75% move  
bytes 2,3 Count for the 25% Tail Shape  
bytes 4,5 Skip count for the Trig Table

Addr	Name	Description
-----	-----	-----
2688	PLN64	P-FFT Length = 64
268E	PLN128	P-FFT Length = 128
2694	PLN256	P-FFT Length = 256
269A	PLN512	P-FFT Length = 512

Then the S-Shape Function

bytes 0,1	Count for the 12.5% Front Shape
bytes 2,3	Cosine Skip Value for the 12.5% Front
bytes 4,5	Count for the 62.5% Middle
bytes 6,7	Count for the 25% Tail Shape
bytes 8,9	Cosine Skip Value for the 25% Tail

Addr	Name	Description
26A0	SLN128	S-FFT Length = 128
26AA	SLN256	S-FFT Length = 256
26B4	SLN512	S-FFT Length = 512

### Trig Table Directory

Addr	Name	Description
26BE	TRGDIR	Trig Directory (with next two words)
27FE	TRGLWA	Trig Last Word Address (Value = #27FE)
26FE	TRGFWA	Trig First Word Address (Value = #26FE)

### Bit Table Mask

The Bit Table Mask is used in the FFT Algorithm  
and consists of ten entries  
#0400, #0200, ..., #0080, #0040, #0000

Addr	Name	Description
26C2	BITTBL	Bit Table Mask

### FFT Length Tables

These Tables are used during the FFT Algorithm

bytes 0,1	Pointer to the Bit Mask Table
bytes 2,3	Shift Count for the BITREV Routine
bytes 4,5	Extra (unused) spaces
bytes 6,7	Extra (unused) spaces
bytes 8,9	Extra (unused) spaces

Addr	Name	Description
26D6	FLN512	FFT Length = 512
26DE	FLN256	FFT Length = 256
26E6	FLN128	FFT Length = 128
26EE	FLN64	FFT Length = 64

## 4.7 Tables

Tables are used for fast data availability and accuracy.

### 4.7.1 FFT/FIT Tables

#### 4.7.1.1 TRGTBL (Trig Table)

TRGTBL is a listing of hex sine values for the first quarter of the sine wave. That is:

$$\sin(2\pi n/512) \quad \text{for } n = 0 \text{ to } 128$$

Values range from 0 (#0000) to almost +1 (#7FFF). It is at addresses #26FE to #27FF. It is used for the SHAPE and FFT functions.

#### 4.7.1.2 LOGTBL (Logarithm Table)

This is a table of Mantissa values, Base 2, for:

$$\log(1 + (n/128) + (1/256))$$

(Note: The 1/256 factor yields the median for the increment from one n to the next n). It resides at addresses #2D00 to #2DFF.

#### 4.7.1.3 INSFIX (Instrument Correction Table)

The correction table is a table of 256 possible frequency entries from DC to 50 Hz to correct for the geophone's filtering characteristics (run at 100 samples per second). It is stored at addresses #2E00 to #2FFF. For more information see the CKFMAX program explanation (Section 5.2.3).

#### 4.7.2 Run Time Tables

SINWAV, a listing of a full cycle sine wave, is used to test the FFT/FIT operation. In addition two listings of earthquake data, PQUAKE and SQUAKE, are stored at locations 7A00 and 7B00 respectively. Trace Programs (Section 6.5) are used to load these into the FIFOs.

Addr	Name	Description
4000	SINWAV	Sine Wave (to #41FF)
7A00	PQUAKE	P-Quake Data (to #7A7F)
7B00	SQUAKE	S-Quake Data (to #7BFF)

## 5.0 FFT/FIT PROGRAM DESCRIPTION

### 5.0.1 FFTMN (FFT Main)

#### 5.0.1.1 Function

The FFTMN program is the master program for FFT/FIT processing. It is called by the WORKER and has two entry points, one for a P-FFT, and one for a S-FFT. Upon transmitting the results to the BOSS it returns control to the WORKER.

#### 5.0.1.2 Calling Program

FTTMN is called by WORKER.

#### 5.0.1.3 Inputs

There are few inputs for this program. Returned from the CKFMAX (Check FMAX) is a flag signifying FMAX validity. There are also the FIT values (see the Internal Variables Section following).

#### 5.0.1.4 Internal Variables

The following are actually FIT Global Variables but are maintained in the programs as Internal Variables. They are being calculated for transmittal to BOSS.

Addr	Name	Description
----	-----	-----
30E8	LNFFT	Length of FFT
30EA	F0	F0, Corner Frequency
30EC	GMAINT	GAMMA Integer
30EE	GMAFRC	GAMMA Fraction
30F0	MXNINT	Maximum Energy Integer
30F2	MXNEXP	Maximum Energy Exponent



#### 5.0.1.5 Subroutines Used

All subroutine programs are called from this program. In addition BMESS (Buss Message Formation) and XBUSS (Transmit Buss) routines are utilized for transmitting the results to the BOSS.

#### 5.0.1.6 Output

The FIT variables listed above are transmitted to the BOSS.

#### 5.0.1.7 Description

Upon entry it calls the subprograms uniquely (for P-FFT or S-FFT) for the LNSET (Length Set), SCALE and SHAPE routines, and then in common for the remaining programs: FFT, UNSCRM, ORDER, MAGAPX, SMTGMX, CKFMAX, GMACPT, and FOLPL.

At the end of the computations the results are sent to the BOSS using the BMESS and XBUSS transmit routines. First one byte is sent to identify the message: either a "F" for a P-FFT or a "G" for a S-FFT. Then the six words of information are sent.

It should be mentioned that there are two potential scenarios where impossible data will work as a flag for out of bounds results. First, if the envelope did not result in a usable spectra (the FMAX, Frequency of Maximum Energy, value is too close to DC or too high a frequency), the FFTMN program will skip GMACPT (Gamma Compute) and FOLPL (FO and Long Period Level) computations and return with "F" or "G", length of the FFT, FMAX (not FO), MXNINT and MXNEXP correct, and zeros for GMAINT and GMAFRC (that is, as error flags). Secondly, if the GMACPT routine results in an invalid result (GAMMA

ASP FFT Program  
FFT/FIT Program Description

less than 1.0) the FOLPL computation will be skipped and the length of the FFT, FMAX, MXINT and MXPX results returned as above. The out of range GAMMA will be its own flag.

There are three patches in this routine. The first and last patches are fixes for a problem involving the SHOLD value with the P-FFTs. SHOLD is the value that allows or prevents the WORKER from logging seismographic entries into the YSFIFO. For the P-FFT the YSFIFO is used for the in-place computation and thus SHOLD should be set to prevent data overwrite. This was not done in the WORKER and must be set, and later cleared here. The middle patch is necessary to cover a programming error for flag passing for CKFMAX results. The patches were left in because of a lack of time for recoding.

## 5.1 FFT Programs

### 5.1.1 LNSET (Length Set)

#### 5.1.1.1 Function

This function sets the P-FFT flag, the Length Values, the Length Pointers, and the BASE value for the FFT routine. If a P-FFT has been called and there has been no S-Wave detection then the default is set:

P-LENGTH = 128

Otherwise the rules are based on the time difference of arrival of the P- and S-Waves (DT).

DT <= 128	P-LENGTH = 64	S-LENGTH = 128
128 < DT <= 600	P-LENGTH = 128	S-LENGTH = 256
600 < DT <= 32K	P-LENGTH = 256	S-LENGTH = 512

#### 5.1.1.2 Calling Program

LNSET is called by FFTMN. There are two entry points, one for a PLNSET (P-Length Set) and one for a SLNSET (S-Length Set).

#### 5.1.1.3 Inputs

The input values are DT, YP- or YS-OLDEST (the arrival time of the Wave), and SLAST (a flag for whether an S-Wave has been detected).

#### 5.1.1.4 Internal Variables

There are no Internal Variables.

#### 5.1.1.5 Subroutines Used

No subroutines are used.

#### 5.1.1.6 Outputs

This function sets the PFTFLG (P-FFT Flag), the Length Values, the Length Pointers, and the BASE value for the FFT routine.

#### 5.1.1.7 Description

The first task for either a P or a S FFT is to check the OLDEST pointer for having been run past the end of the FIFO (to LWA+1). This could occur because of a difference in testing for out of bounds addresses between the WORKER and the FFT programs (testing before vs. after use, see the Problems Section 6.6). The pointer is reset to the FIFO FWA if necessary.

The P-FFT flag is set true for a P-FFT, false for a S-FFT. The Length Values and Pointers are then set based upon the rules. The BASE value is set to the First Word Address of the data to be FFT'd. If it is a P-FFT the data will be moved to the YSFIFO, thus BASE = YSFWA. If it is a S-FFT the FFT will be done In-Place, thus BASE = YS-OLDEST. There are four sections to the program, as listed:

PLNSET: (P-Length Set)

Check for YP-OLDEST at YPLWA+1, If past end  
Then point YP-OLDEST at YPFWA

Set PFTFLG True (#FF)

Check for an S-Wave, if SLAST <> #0000 (Flag for an S-Wave)  
Then Goto DTCHK  
Else (No S-Wave) Point to P-FFT = 128, Goto STPSLN

SLNSET: (S-Length Set)

Check for YS-OLDEST at YSLWA+1, If past end  
Then point YS-OLDEST at YSFWA

Set PFTFLG False (#00)

DTCHK: (DT Check)

Get DT into R(ZAP3)

Test DT, and increment pointer, until <= Boundaries

STPSLN: (Store P-Length and S-Length Pointers)

Store Length Values and Pointers

If P-FFT

Then set BASE = YSFWA

Else set BASE = YS-OLDEST

Exit



## 5.1.2 SCALE

### 5.1.2.1 Function

This routine scales the P-FFT or the S-FFT data in the FIFO to the maximum via the use of the DTASCL (Data Scale) subroutine.

### 5.1.2.2 Calling Program

SCALE is called by FFTMN. There are two entry points, PSCALE for a P-FFT, and SSCALE for an S-FFT.

### 5.1.2.3 Inputs

The primary input is the time series data in the FIFO. Additional inputs are the P- or S-FFT Length and the YP- or YS-OLDEST address.

### 5.1.2.4 Internal Variables

There are no Internal Variables.

### 5.1.2.5 Subroutines Used

DTASCL (Data Scale) is used to scale the FIFO Data to the maximum values.

### 5.1.2.6 Output

The outputs are the data in the FIFO, correctly scaled, TOTLSH (Total Shift Count) and, if it was a P-FFT, PSHIFT (P-Shift Count).

### 5.1.2.7 Description

The SCALE routine initializes the TOTLSH (Total Shift Counter) to zero and then sets the registers to call the DTASCL (Data Scale) subroutine, allowing the maximum scaling to #7FFF (positive) or #8000 (negative). Upon return, if the call had been for a P-FFT, the TOTLSH value is passed to the PSHIFT (P-FIFO Shift Value) register. (Note: This is the only time a scaling occurs in the YPFIFO, and is the reason for keeping special track of the shift value.

ASP FFT Program  
FFT/FIT Program Description

5.1.3 SHAPE

5.1.3.1 Function

A Windowing Function needs to be applied to the Time Domain data prior to performing the FFT to minimize the Frequency Leakage inherent in a Sampled Data system. The Shaping routine shapes the data by the Raised Cosine Bell function, that is:

$$\text{Output} = \text{Input} * \{1/2 * [1 - \cos(2*\pi*n/\text{Length FFT})]\}$$

This is done for 12.5% of the front of the YS-FIFO and for 25% of the tail of both the YP- and YS-FIFO. (These values were set by the seismologists after extensive simulation on the LBL mainframe). Also at this time the YPFIFO data will be moved to the YSFIFO if a P-FFT has been called.

5.1.3.2 Calling Program

SHAPE is called by FFTMN. There are two entry points, PSHAPE for a P-FFT, and SSHAPE for an S-FFT.

5.1.3.3 Inputs

The primary input is the time component data in the FIFO. Additional inputs are the P- or S-FFT Length and the YP- or YS-OLDEST address.

#### 5.1.3.4 Internal Variables

CSPTR (Cosine Pointer) is used to address the Cosine value as it is incremented through the Cosine table.

Addr	Name	Description
30EC	CSPTR	Cosine Pointer

#### 5.1.3.5 Subroutines Used

The P-SHAPE routine uses the BDRYST (Boundary Set) and BDRYCK (Boundary Check) subroutines for the 75% move. The BDRYST is reinitialized for the 25% shape. The SHPSET (Shape Set) routine sets up the cosine value, incremented by a table skip count. The multiplication is done by CSMULT (Cosine Multiply). Values for the indices are set from the P-SHAPE tables, via PLNPTR (P-Length Pointer).

The S-SHAPE routine also calls BDRYST and BDRYCK, twice, once for each shape. SHPSET and CSMULT are used, with new table SKIP counts. The new entry in this case is the 67.5% center skip count (the amount of the FIFO left unshaped). S-SHAPE tables are pointed to by SLNPTR (S-Length Pointer).

#### 5.1.3.6 Output

The only output is the time series data in the YSFIFO, shaped by the raised cosine bell function.

### 5.1.3.7 Description

The algorithms are slightly different for the P-FFT than for the S-FFT. For the P-FFT the data is moved from the YPFIFO to the YSFIFO. The first 75% of the data is moved untouched. The last 25% is shaped with the raised cosine bell  $[1/2 - 1/2(\cos(2\pi n/N))]$ . For the S-FFT the operation is done in-place and the first 12.5% and the last 25% are shaped with the center 62.5% left untouched. The program is as follows:

#### PSHAPE: (P-FFT Shape)

Initialize Pointers

Call BDRYST for Move BDRYOK (Boundary OK) and FREELN (Free Length)

#### 75% Move

Move 75% of YPFIFO data to YSFIFO, Checking for Address Wraparound  
(If Address = YPLWA+1  
Then Address = YPFWA)

#### 25% Shape

Call BDRYST for Shape BDRYOK and FREELN

Get Pointer to Cosine Value

#### GTCSVL: (Get Cosine Value)

Call SHPSET (Shape Set) to get Cosine Value into CSVLU

Calculate  $[(-1/2 \cos) + 1/2]$

Multiply by YPFIFO Data, Store into YSFIFO

Call BDRYCK to Test for Wraparound, if Address = YPLWA+1  
Then Set Address = YPFWA

If Not Done,  
Then Goto GTCSVL

Exit



SSHAPE: (S-FFT Shape)

Initialize Pointers

Call BDRYST for 12.5% Shape BDRYOK and FREELN

Get Pointer to Cosine Value

SHDSHP: (S-FFT Head Shape)

Call SHPSET (Shape Set) to get Cosine Value into CSVLU

Calculate  $[(-1/2 \cos) + 1/2]$

Multiply by YSFIFO Data, Store into YSFIFO

Call BDRYCK to Test for Wraparound, if Address = YSLWA+1  
Then Set Address = YSFWA

If Not Done,  
Then Goto SHDSHP

62.5% Skip

Add 62.5% Count to Pointer

Check for Address Wraparound, If > YSLWA+1  
Then Set past YSFWA

25% Shape

Call BDRYST for 25% Shape BDRYOK and FREELN

Get Pointer to Cosine Value

STLMTH: (S-FFT Tail Shape Math)

Call SHPSET (Shape Set) to get Cosine Value into CSVLU

Calculate  $[(-1/2 \cos) + 1/2]$

Multiply by YSFIFO Data, Store into YSFIFO

Call BDRYCK to Test for Wraparound, if Address = YSLWA+1  
Then Set Address = YSFWA

If Not Done,  
Then Goto STLMTH

Exit

#### 5.1.4 FFT

##### 5.1.4.1 Function

The real data will be treated as if it were complex data (Real and Imaginary). This will require the Reordering function to be run after the completion of the FFT (and after the Unscrambling).

The heart of the FFT algorithm is the generation of new data columns from the old columns by the equations:

$$\begin{aligned} Y_n(k) &= Y_o(k) + [W(p) * Y_o(k+(N/2))] \\ Y_n(k+(N/2)) &= Y_o(k) - [W(p) * Y_o(k+(N/2))] \end{aligned}$$

Where:

$Y_n$  is the New Column  
 $Y_o$  is the Old Column (both Complex Data)  
 $W(p)$  is  $[\cos(2\pi p/N) + j\sin(2\pi p/N)]$  (also Complex)

And:

$k$  and  $p$  are index values  
 $N$  is the length of the FFT

Expanding the Complex Data to Real and Imaginary components, written in the Algorithm Nomenclature:

$$\begin{aligned} M(\text{LOREAL}) &= M(\text{LOREAL}) + \text{TREAL} \\ M(\text{LOIMAG}) &= M(\text{LOIMAG}) + \text{TIMAG} \\ M(\text{HIREAL}) &= M(\text{LOREAL}) - \text{TREAL} \\ M(\text{HIIMAG}) &= M(\text{LOIMAG}) - \text{TIMAG} \end{aligned}$$

These are the Quad Values referred to below, where:

$$\begin{aligned} \text{TREAL} &= [M(\text{HIREAL}) * \text{CSVLU}] + [M(\text{HIIMAG}) * M(\text{SNPTR})] \\ \text{TIMAG} &= [M(\text{HIIMAG}) * \text{CSVLU}] + [M(\text{HIREAL}) * M(\text{SNPTR})] \end{aligned}$$

These are the Temporary Values referred to below.

This is computed for the 0 column, then for the 1 column, then for the 2 column, etc. up to the E column where  $N = 2^{(E-2)}$ . (The FFT is done on pairs of complex data at a time, that is four values, thus the  $2^2$  reduction in the number of columns).

#### 5.1.4.2 Calling Program

FFT is called by FFTMN.

#### 5.1.4.3 Inputs

The primary input is (modified) time series data in the FIFO. Additional inputs are FLNPTR, with the FFT Length, PFTFLG, the PFFT/SFFT Flag, and the BASE address of the data in the FIFO.

#### 5.1.4.4 Internal Variables

Addr	Name	Description
30E0	COLPTR	Column Pointer
30E2	NDBTPT	End Bit Pointer
30E4	CPBTPT	Column Bit Pointer
30E6	ENDBIT	End Bit Value
30E8	COLBIT	Column Bit Value
30EA	TRGCTR	Trig Counter Value
30EC	CSPTR	Cosine Pointer Value
30EE	SNPTR	Sine Pointer Value
30F0	TRGPTR	Trig Pointer Value
30F2	LOREAL	Low Real Address Pointer
30F4	LOIMAG	Low Imaginary Address Pointer
30F6	HIREAL	High Real Address Pointer
30F8	HIIMAG	High Imaginary Address Pointer
30FA	TREAL	Temporary Real Value
30FC	TIMAG	Temporary Imaginary Value

How they are used is described below:

CLBTPT (Column Bit Pointer)

Points to Bit Mask Table

Starts at size of FFT ( $512 = \#0400$ ,  $256 = \#0200$ ,  $128 = \#0100$ ,  
 $64 = \#0080$ )

ASP FFT Program  
FFT/FIT Program Description

At start of column goes to next Bit Mask Table entry  
(#0400 to #0200, #0200 to #0100, #0100 to #0080, #0080 to #0000)  
If new value = #0000, then FFT done.

COLBIT (Column Bit)

Receives value from Bit Mask Table  
Used to compute high pair of new "Quads" address  
Helps see if element of pair done (see COLPTR)  
Updates COLPTR if pair done.

COLPTR (Column Pointer)

At new column starts at #0000  
Points at each new element of array  
(updated by 4 each time to next Real Value)  
Used to compute new quad's address  
Checks to see if pair done (if COLBIT picks "1" in COLPTR  
Then update COLPTR by COLBIT (COLPTR = COLPTR + COLBIT)

NDBTPT (End Bit Pointer)

Points to bit Mask Table (512 = #0400, 256 = #0200, 128 = #0100,  
64 = #0080)  
Value pointed to will mark end of column to updated COLPTR  
Does not change for FFT

ENDBIT (End Bit)

Receives Bit Mask Table value in NDBTPT (End Bit Pointer)  
Marks end of column

The following TRIG variables are passed between the FFT routine and  
the TRGSET (Trig Set) subroutine.

TRGCTR (Trig Counter)

Starts at #0000  
Every time skips over "Done Half" of Pairs it increments by 2

TRGPTR (Trig Pointer)

Bit reversal of TRGCTR  
Used to compute Trig Values

SNPTR, CSPTR (Sine and Cosine Table Pointers)

Set by TRGSET  
SNPTR = TRGFWA + TRGPTR (Trig Table First Word Address + Pointer)  
CSPTR = TRGLWA - TRGPTR (Trig Table Last Word Address - Pointer)

LOREAL, LOIMAG, HIREAL, HIIMAG

Low and High, Real and Imaginary Pointers to Quad of Data for  
Butterfly Computations

TREAL, TIMAG

Temporary locations for multiplication results, prior to adding  
and subtracting from previous columns low data pair.

#### 5.1.4.5 Subroutines Used

TRGSET (Trig Set) is used for computing Trig Pointers and Values. DTASCL (Data Scale) is used to scale the FIFO Data to the maximum values. ADDSTF (Add and Stuff) is used to calculate the LOREAL and LOIMAG or the HIREAL and HIIMAG addresses. WRPTST (Wraparound Test) is used to test for Y-FFT address locations past the end of the FIFO. CSMULT and SNMULT (Cosine and Sine Multiply) are used to multiply the values. NEWVAL (New Value) is used to compute the store the new values for LOREAL and HIREAL or LOIMAG and HIIMAG.

#### 5.1.4.6 Output

The only output is the data in the FIFO, now modified as frequency components, but yet to be Unscrambled and Reordered.

#### 5.1.4.7 Description

The FFT routine is quite involved. A lot of effort has gone into optimizing it for speed by taking advantage of the symmetries available at the bit level. There are nine sections and they proceed as follows:

BIT/PTR SET:

Initialize Pointers, Masks, and Counters for Start

NEWCOL: (New Column)

You're now at a new Column

Reinitialize/Increment Pointers, Masks, and Counters

Test for Done, If Done  
Then Goto DUNFFT

Get new Trig Pointers and Values



ASP FFT Program  
FFT/FIT Program Description

MAGNITUDE CHECK:

Check for Maximum Dynamic Range

PRSKCH: (Pair Skip Check)

You're at a new Pair

Check to see if Pair is Last Half of Done Quad, if Not Done  
Then Goto QDADDR

Else get new Quad's Address

Check to see if Column Done, if Done  
then Goto NEWCOL

Get new Trig Pointers and Values

QDADDR: (Quad Address)

Compute new Quad's Addresses

If S-FFT

Then Check for Address Wraparound, Correct if Necessary

TRGMLT:

Compute Temporary Values (TREAL, TIMAG) shown above

ADD TRIG:

Update Quad (LOREAL, LOIMAG, HIREAL, HIIMAG) as above

NEXT QUAD:

Get to Next Quad

Goto PRSKCH:

DUNFFT:

Exit

### 5.1.5 UNSCRM (Unscramble Data)

#### 5.1.5.1 Function

A result of the Decimation-In-Frequency FFT algorithm is that the data is left addressed in Bit-Reversed Scrambled order (see the TRGSET/SHPSET Section 5.3.5.1 for an explanation of the Bit-Reversal operation). This routine Unscrambles the data by address, resulting in correctly Ordered data.

#### 5.1.5.2 Calling Program

UNSCRM is called by FFTMN.

#### 5.1.5.3 Inputs

The primary input is frequency component data in the FIFO. Additional inputs are FLNPTR, with the FFT Length, the ENDCNT, the End Count, PFTFLG, the PFFT/SFFT Flag, and the BASE address of the data in the FIFO.

#### 5.1.5.4 Internal Variables

Addr	Name	Description
30E0	ENDCNT	End Count
30E2	SHFCNT	Shift Count
30E4	NMPTR	Normal Pointer
30E6	SWPTR	Switched Pointer
30E8	NMREAL	Normal Real Index
30EA	NMIMAG	Normal Imaginary Index
30EC	SWREAL	Switched Real Index
30EE	SWIMAG	Switched Imaginary Index

ASP FFT Program  
FFT/FIT Program Description

They are used as follows:

ENDCNT (End Count)

Will contain number to compare against NMPTR for done  
Either #0400, #0200, #0100, or #0080 for FFT Length =  
512, 256, 128, or 64

SHFCNT (Shift Count)

Shifting count for bit-reversal routine  
Either #000C, #000B, #000A, or #0009 for FFT Length =  
512, 256, 128, or 64

NMPTR and SWPTR (Normal and Bit-Reversed Pointers for Arrays)

NMPTR is also tested against ENDCNT for done

NMREAL, NMIMAG, SWREAL, and SWIMAG

Will contain Addresses of Quad Data for Switching Operation

#### 5.1.5.5 Subroutines Used

WRPTST (Wraparound Test) is used to test for Y-FFT address locations past the end of the FIFO. ADDSTF (Add and Stuff) is used to calculate the LOREAL and LOIMAG or the HIREAL and HIIMAG addresses. The UNSCRM routine also contains its own BITREV Internal Subroutine, which performs the Bit Reversal a variable number of shifts (as opposed to the TRGSET Fixed Shift Bit Reversal).

#### 5.1.5.6 Output

The only output is the data in the FIFO, correctly reshuffled.

#### 5.1.5.7 Description

There are five sections of the program, and the Internal Subroutine. They proceed as follows:

INITIALIZE:

Initialize Counters and Pointers

GOSHFT: (Go Shift)

You're at new Address Pair

SWPTR = BITREV(NMPTR)

Check for Quad already Done, If Done  
Then Goto PTRUP

QDADDR: (Quad Addresses)

You're at known good Quad

Compute Address Pointers NMREAL, NMIMAG, SWREAL and SWIMAG

If S-FFT

Then check for Address Wraparound, Correct if Necessary

MEMSWP: (Memory Swap)

Here the data is switched

NMREAL with SWREAL, and

NMIMAG with SWIMAG

PRTUP: (Pointer Up)

Increment Pointer to Next Pair

Check for Done, If Not Done  
Then Goto GOSHFT

Exit

---

BITREV: (Bit Reversal Internal Subroutine)

Shift Normal Pointer into Switched Pointer  
Do SHFTCT (Shift Count) Times

Return via Reset to PC

### 5.1.6 ORDER (Column Reorder)

#### 5.1.6.1 Function

A result of splitting the  $(2*N)$  Real Time Series Data into Two  $N$  Series (Real and Imaginary) and then Transforming the  $N$  Series is that is that the output is ordered as per the Split Series, not the Original. To regain the Original, reorder as follows (see [Brig], pg. 169):

$$\begin{aligned} 2*Xr(n) = & [R(n)+R(N-n)] \\ & + \{\cos[2*\pi*n/(2*N)] * [I(n)+I(N-n)]\} \\ & - \{\sin[2*\pi*n/(2*N)] * [R(n)-R(N-n)]\} \quad (1) \end{aligned}$$

$$\begin{aligned} 2*Xi(n) = & [I(n)-I(N-n)] \\ & - \{\sin[2*\pi*n/(2*N)] * [I(n)+I(N-n)]\} \\ & - \{\cos[2*\pi*n/(2*N)] * [R(n)-R(N-n)]\} \quad (2) \end{aligned}$$

for  $n = [0, 1, \dots, (N-1)]$

#### 5.1.6.2 Calling Program

ORDER is called by FFTMN.

#### 5.1.6.3 Inputs

The primary input is the frequency component data in the FIFO. Additional inputs are FLNPTR, with the FFT Length, PFTFLG, the PFFT/SFFT Flag, and the BASE address of the data in the FIFO.



#### 5.1.6.4 Internal Variables

The following Internal Variables are used in the routine:

Addr	Name	Description
30E0	LOREAL	Low Real Address Pointer
30E2	LOIMAG	Low Imaginary Address Pointer
30E4	HIREAL	High Real Address Pointer
30E6	HIIMAG	High Imaginary Address Pointer
30E8	TRGINC	Trig Increment
30EA	LOSIN	Low Sine Value
30EC	LOCOS	Low Cosine Value
30EE	RLADD	Real Add
30F0	RLSBLH	Real Sub Low - High
30F2	RLSBHL	Real Sub High - Low
30F4	IMADD	Imaginary Add
30F6	IMSBHL	Imaginary Sub Low - High
30F8	IMSBHL	Imaginary Sub High - Low

#### 5.1.6.5 Subroutines Used

WRPTST (Wraparound Test) is used to test for Y-FFT address locations past the end of the FIFO. DTASCL (Data Scale) is used to scale the FIFO Data to the maximum values. The ORDER routine also contains two Internal Subroutines; COMBO for doing LO/HI-REAL/IMAG math and storage, and ORDMLT, the ORDER multiply.

#### 5.1.6.6 Output

The only output is the data in the FIFO, now correctly organized as Real and Imaginary components.

### 5.1.6.7 Description

The Column Reorder routine proved to be a complicated program. It is quite involved, as it does not have the symmetry which characterized the FFT. It is therefore very large (the body of the program is 609 bytes, versus 372 bytes for the FFT).

In order to simplify the trigonometric computations it is recognized that trig entries for  $n$  and  $(N - n)$  are equal for the sine function and opposite for the cosine function. Substituting  $(N - n)$  for  $n$ , equations (1) and (2) may be rewritten:

$$\begin{aligned} 2 * X_r(N-n) = & [R(N-n) + R(n)] \\ & + \{ \cos[2 * \pi(N-n)/(2 * N)] * [I(N-n) + I(n)] \} \\ & - \{ \sin[2 * \pi(N-n)/(2 * N)] * [R(N-n) + R(n)] \} \end{aligned} \quad (3)$$

$$\begin{aligned} 2 * X_i(N-n) = & [I(N-n) + I(n)] \\ & - \{ \sin[2 * \pi(N-n)/(2 * N)] * [I(N-n) + I(n)] \} \\ & - \{ \cos[2 * \pi(N-n)/(2 * N)] * [R(N-n) + R(n)] \} \end{aligned} \quad (4)$$

Using the facts that:

$$\begin{aligned} \sin[\pi(N-n)/N] &= \sin[\pi(n)/N]; \\ \cos[\pi(N-n)/N] &= -\cos[\pi(n)/N]; \text{ and} \\ [R(N-n) - R(n)] &= -[R(n) - R(N-n)] \end{aligned}$$

Equations (3) and (4) can be rewritten:

$$\begin{aligned} 2 * X_r(N-n) = & [R(n) + R(N-n)] \\ & - \{ \cos[2 * \pi * n/(2 * N)] * [I(n) + I(N-n)] \} \\ & + \{ \sin[2 * \pi * n/(2 * N)] * [R(n) - R(N-n)] \} \end{aligned} \quad (5)$$

$$\begin{aligned} 2 * X_i(N-n) = & [I(N-n) - I(n)] \\ & - \{ \sin[2 * \pi * n/(2 * N)] * [I(n) + I(N-n)] \} \\ & - \{ \cos[2 * \pi * n/(2 * N)] * [R(n) - R(N-n)] \} \end{aligned} \quad (6)$$

Now grouping these equations around their common trig factors, using equations (1) and (2) for the lower components and equations (5) and (6) for the higher components, they can be rewritten in a general sense, using the following program variables. (The data will be allowed to be scaled by 2; this factor will be included later in the SMTGMX program - see Section 5.2.2.7).

```

LOREAL = RLADD + (LOCOS * IMADD) - (LOSIN * RLSBLH) (7)
LOIMAG = IMSBLH - (LOSIN * IMADD) - (LOCOS * RLSBLH) (8)
HIREAL = RLADD - (LOCOS * IMADD) + (LOSIN * RLSBLH) (9)
HIIMAG = IMSBLH - (LOSIN * IMADD) - (LOCOS * RLSBLH) (10)

```

Where, for  $n = 1, 2, \dots, [(N/2)-1]$ :

```

LOCOS = cos[2*pi*n/(2*N)]
LOSIN = sin[2*pi*n/(2*N)]
LOREAL = R(n)
LOIMAG = I(n)
HIREAL = R(N-n)
HIIMAG = I(N-n)
RLADD = LOREAL + HIREAL
RLSBLH = LOREAL - HIREAL
RLSBHL = HIREAL - LOREAL
IMADD = LOIMAG + HIIMAG
IMSBLH = LOIMAG - HIIMAG
IMSBHL = HIIMAG - LOIMAG

```

These variables are used in the routine. In addition there is a TRGINC value, used for moving through the Trig Table.

It should be noted that there are two special cases not covered correctly by equations (7) to (10). From equations (1) and (2), for  $n = 0$ :

$$\begin{aligned}
 2Xr(0) = & \quad [R(0)+R(N)] \\
 & + \{ \cos[0] \quad * [I(0)+I(N)] \} \\
 & - \{ \sin[0] \quad * [R(0)-R(N)] \} \quad (11)
 \end{aligned}$$

$$\begin{aligned}
 2Xi(0) = & \quad [I(0)-I(N)] \\
 & - \{ \sin[0] \quad * [I(0)+I(N)] \} \\
 & - \{ \cos[0] \quad * [R(0)-R(N)] \} \quad (12)
 \end{aligned}$$

ASP FFT Program  
FFT/FIT Program Description

Therefore, remembering  $R(0) = R(N)$  and  $I(0) = I(N)$ :

$$\begin{aligned} 2*Xr(0) &= 2[R(0)] + 2[I(0)] \\ &= 2[R(0) + I(0)] \end{aligned} \quad (13)$$

$$2*Xi(0) = 0 \quad (14)$$

This is not the result which is obtained from the generalized equations. They will be covered uniquely in the program, prior to the general function. The other exception, for  $n = (N/2)$ , will be explained after the program description.

The routine has ten sections and two subroutines. Control is as listed:

INITIALIZE:

Initialize Trig and FIFO Pointers and Increment Values

SCALE DATA:

Scale Data for Maximum Dynamic Range

SET ADDRESSES:

Set Addresses LOREAL, LOIMAG, HIREAL and HIIMAG

Check for Address Wraparound, Correct if Necessary

CMPZRO: (Compute Zero Values)

Compute LOREAL(0) and LOIMAG(0) Values and Store

Goto LOUP

MIDSIX: (Compute Six Middle Values)

Compute RLADD, RLSBLH, RLSBHL, IMADD, IMSBLH and IMSBHL values

STORE VALUES:

Compute and Store the Final Values LOREAL, LOIMAG, HIREAL and HIIMAG

INCREMENT/DECREMENT TRIG POINTERS:

Increment/Decrement Trig Pointers

Check for Done, If Done  
Then Goto FIXHLF

LOUP: (LOREAL and LOIMAG FIFO Pointers Up)

Increment/Decrement FIFO Pointers

Compute new LOREAL, LOIMAG, HIREAL and HIIMAG Addresses

If S-FFT  
Then check for Address Wraparound, Correct if Necessary

Goto MIDSIX

FIXHLF: (Fix Half Values)

Divide  $I(N/2)$  by two for Correction

NDODRD: (End Order)

Exit

---

COMBO: (Combine LO/HI-REAL/IMAG Internal Subroutine)

Compute HI-LO, LO-HI, and LO+HI Values

Store Them

Return via Reset to PC

ORDMLT: (Order Multiply Internal Subroutine)

Compute  $M(M(R(ZAP1))) * M(R(MQ)) \rightarrow R(ZAP3)$

Return via Reset to PC



ASP FFT Program  
 FFT/FIT Program Description

As mentioned, the FIXHLF routine corrects the values for the  $N/2$  entries. From equations (1) and (2), for  $n = N/2$ :

$$\begin{aligned} 2*Xr(N/2) = & [R(N/2)+R(N-(N/2))] \\ & + \{\cos[2*\pi*(N/2)/(2*N)] * [I(N/2)+I(N-(N/2))]\} \\ & - \{\sin[2*\pi*(N/2)/(2*N)] * [R(N/2)-R(N-(N/2))]\} \end{aligned} \quad (15)$$

$$\begin{aligned} 2*Xi(N/2) = & [I(N/2)-I(N-(N/2))] \\ & - \{\sin[2*\pi*(N/2)/(2*N)] * [I(N/2)+I(N-(N/2))]\} \\ & - \{\cos[2*\pi*(N/2)/(2*N)] * [R(N/2)-R(N-(N/2))]\} \end{aligned} \quad (16)$$

Since  $\cos(\pi/2) = [R(N/2)-R(N/2)] = [I(N/2)-I(N/2)] = 0$ , and  $\sin(\pi/2) = 1$ , equations (15) and (16) become:

$$2*Xr(N/2) = 2[R(N/2)] \quad (17)$$

$$2*Xi(N/2) = -2[I(N/2)] \quad (18)$$

As computed in the routine for  $n = (N/2)$  case, "REAL(N/2)" is left with  $2*["REAL"]$ , the correct entry (almost by luck). The result for "IMAG(N/2)" is  $-4*["IMAG"]$ . As seen from above (18) this is twice as large as desired, thus the correction. (Note: this correction after the fact was determined to be the best way to implement this non-general calculation, in that the address pointers were in-place automatically. It may not be pretty but it works).

## 5.2 FIT Programs

What is desired of the FIT Programs is to fit the P- or S-Wave Spectra for the amplitude response of a V-pole low-pass Butterworth filter:

$$A = \frac{A_0}{[1 + (f/f_0)^2]^{0.5}}$$

Figure 5.2.a shows the computation implemented for two separate events. Shown are the original time series  $x(t_i)$ , the demeaned time series  $x'(t_i)$ , PT and ST arrival times, and the resulting P- and S-Wave spectra. From this can be seen the fitted filter response for Long Period Level (LPL), Corner Frequency ( $f_0$ ), and High Frequency Slope ( $\gamma$ ).

The required operations will be implemented in the following order, via the referenced routines:

MAGAPX: (Magnitude Approximate)

- (1) Approximate the Magnitude of the Frequency Components from the Real and Imaginary values.

SMTGMX: (Smooth and Tag Maximum)

- (2) Smooth the Magnitude Function with an Eight-Point Average.
- (3) Select the Maximum Magnitude Value. Save the Index as FMAX, and the Value as MXVALU and MXEXP.

CKFMAX: (Check FMAX)

- (4) Correct the Maximum Value for Geophone Frequency Response, Geophone Gain and Amplifier Gain. Save as MXNINT and MXNEXP.
- (5) Check for Valid Event, i.e. verify FMAX Index not too Low nor too High. Exit with OK or Error Flag.

GMACPT: (GAMMA Compute)

- (6) Compute spectra High Frequency Slope.

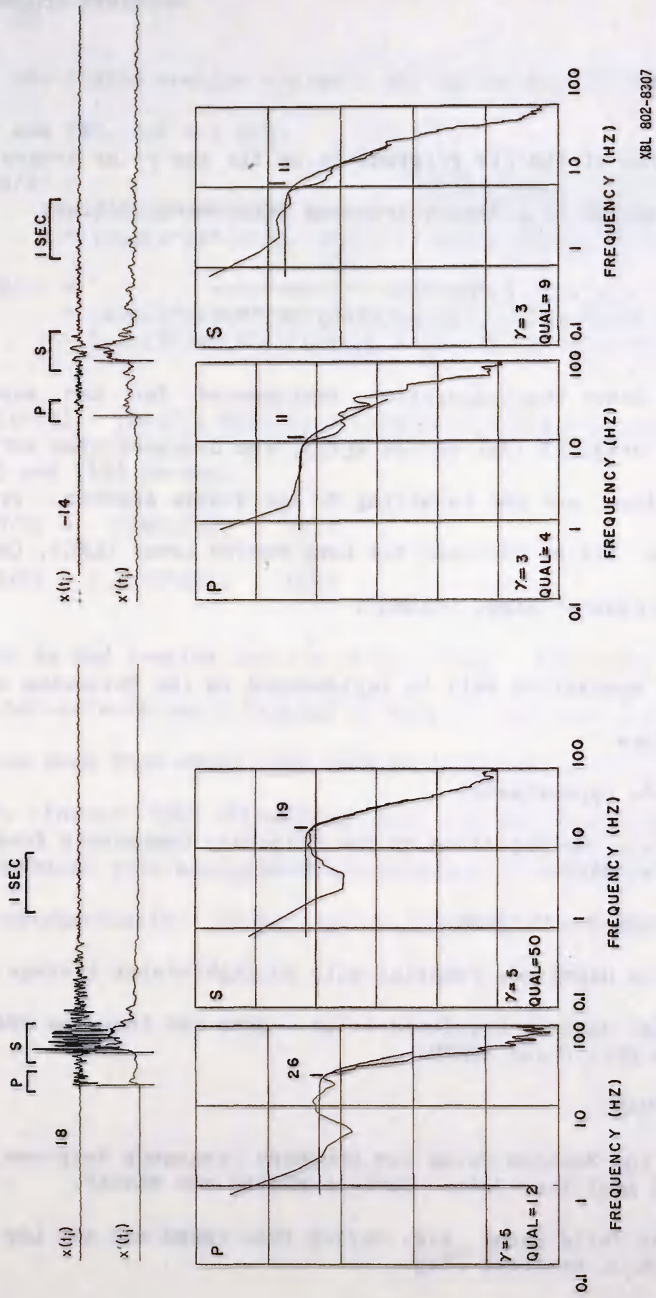


Figure 5.2.a Examples of ASP Processing for Two Events

FOLPL: (FO and Long Period Level Calculation)

- (7) Check for valid GAMMA, return with error Flag if out-of-bounds.  
Calculate FO and Long Period Level based upon FMAX index and spectra value.

5.2.1 MAGAPX (Magnitude Approximate)

5.2.1.1 Function

For the fitting program the data will required to be in the Magnitude/Phase form. (Phase information is not required at this time and will be stored as "0"). The magnitude is of course:

$$\text{Magnitude} = [(R^2) + (I^2)]^{(1/2)}$$

In a microprocessor however a square root algorithm is a non-trivial function (especially without divide hardware), and requires some memory and a lot of time. It has thus been determined by the seismologists that an approximation routine is accurate enough for the fitting program. The approximation that will be implemented is:

Where R(n) Real Data Entry  
I(n) Imaginary Data Entry  
M(n) Magnitude Approximation  
Abs Absolute Value

and G(n) = Greatest [Abs[R(n)] or Abs[I(n)]  
L(n) = Least [Abs[R(n)] or Abs[I(n)]

If  $G(n)/4 \Rightarrow L(n)$

Then  $M(n) = G(n)$   
Else  $M(n) = 0.7659 * [G(n) + L(n)]$

(Note: 0.7659 is the average  $[1/(\sin + \cos)]$  for  $\arctan(1/4)$  and  $\arctan(1/1)$ ).

#### 5.2.1.2 Calling Program

MAGAPX is called by FFTMN.

#### 5.2.1.3 Inputs

The primary input is the frequency component data in the FIFO. Additional inputs are FLNPTR, with the FFT Length, and the BASE address of the data in the FIFO. There is an Internal Constant, the Approximation Factor, 0.7659 (#6209).

#### 5.2.1.4 Internal Variables

There are no Internal Variables.

#### 5.2.1.5 Subroutines Used

The MAGAPX routine uses the BDRYST (Boundary Set) and BDRYCK (Boundary Check) subroutines for checking for Address Wraparound. Multiplication is done by SNMULT. DTASCL (Data Scale) is used to scale the FIFO Data to the maximum values.

#### 5.2.1.6 Output

The only output is the data in the FIFO. The first entry of the "n" value pair will be the Magnitude Approximation. The second entry will be zeroed (so as to not disrupt further Scaling Operations).



### 5.2.1.7 Description

In the following description the "(n)" notation will be suppressed for clarity. There are seven sections to the code. They proceed as follows:

#### SCALE DATA:

Scale Data for Maximum Range

#### INITIALIZE:

Initialize Pointers

Call BDRYST for BDRYOK (Boundary OK) and FREELN (Free Length) values

Setup for Magnitude Approximation

#### NEXTMN: (Next Magnitude Approximation)

Calculate Absolute Value of R, Abs(R)

Check for Address Wraparound, Reset if Necessary

Calculate Absolute Value of I, Abs(I)

#### CMPRRI: (Compare Abs(R) and Abs(I))

If Abs(I) > Abs(R)  
Then Goto IGTR  
Else Abs(R) => Abs(I)

If Abs(R)/4 < Abs(I)  
Then Goto FCTXSM  
Else Abs(R)/4 => Abs(I), Goto NDDNCK  
(Magnitude is as it should be, in R)

#### IGTR: (Abs(I) Greater Than Abs(R))

If Abs(I)/4 < Abs(R)  
Then Goto FCTXSM  
Else Abs(I)/4 => Abs(R), Store Abs(I) into R, Goto NDDNCK

#### FCTXSM: (Factor Times Sum)

Magnitude =  $0.7659 * (Abs(R) + Abs(I))$ , Store into R Location

ASP FFT Program  
FFT/FIT Program Description

NDDNCK:

Store #0000 into I Location

Call BDRYCK (Boundary Check) to Test/Correct Address Wraparound

If Not Done, Goto NEXTMN

Exit

### 5.2.2 SMTGMX (Smooth and Tag Maximum)

#### 5.2.2.1 Function

This routine uses a running 8-point average to smooth the frequency components, and tags the greatest value (after smoothing). It assumes that the magnitude values are in every other location in the FIFO, starting with the BASE address, with entries of #0000 in between. It returns with the smoothed values replacing the original magnitude entries, and:

MXADDR containing the Address of the Maximum Value;  
FMAX the Frequency Index of the Maximum Value;  
MXVALU the (Fixed Point Fraction) Maximum Value; and  
MXEXP the Exponent (Shift) Value of the Maximum.

Note: The Maximum Value was detected prior to modification by the Instrument Correction Table (done in CKFMAX). This was the method selected by the LBL Geologists.

#### 5.2.2.2 Calling Program

SMTGMX is called by FFTMN.

#### 5.2.2.3 Inputs

The primary input is frequency component data in the FIFO, stored as Magnitude values in the "even" locations, with #0000 in the "odd" locations. Additional inputs are FLNPTR, with the FFT Length, and the BASE address of the data in the FIFO.

#### 5.2.2.4 Internal Variables

There are no Internal Variables. Note: The returned values listed above are actually some of the FIT Global Variables. They are listed in the code as Internal but since they are maintained by all code running afterward (including FFTMN) they are actually Global.

#### 5.2.2.5 Subroutines Used

The SMTGMX routine uses the BDRYST (Boundary Set) subroutine to initialize to check for for Address Wraparound. The Boundary Checking is done by SMBDCK (Smooth Boundary Check), an Internal Subroutine necessary because two pointers need to be checked (see Section 5.2.2.7). DTASCL (Data Scale) is used to scale the FIFO Data to the maximum values.

#### 5.2.2.6 Output

One of the outputs is the data in the FIFO, smoothed as described. The other outputs are the calculated values, in the following locations:

Addr	Name	Description
----	-----	-----
30E0	MXADDR	Address of Maximum
30E2	FMAX	Frequency Index of Maximum
30E4	MXVALU	Maximum Value (Integer)
30E6	MXEXP	Maximum Value (Exponent "Shift" Value)

### 5.2.2.7 Description

The operation which will be performed in this program is:

$$S(n) = [M(n-3) + \dots + M(n) + \dots + M(n+4)]/8, \text{ for}$$

$$n = 3, 4, \dots, (\text{Spectra Length} - 5)$$

Where:  $S(n)$  is the Smoothed Magnitude Value

$M(n)$  is the Approximated Magnitude Value, and the

Spectra Length is either 32, 64, 128, or 256

The program will also find the Maximum Spectra Value (or the last occurrence if two or more are equal). The returns are the MXADDR, FMAX, MXVALU, and MXEXP, as listed above.

It should also be noted that in order to perform the eight element sum for the running window two special constructs are used. First a pair of pointers, one for the head, and one for the tail, are necessary. This also requires a unique SMBDCK (Smooth Boundary Check) subroutine to test both pointers. Second, a 24-bit accumulator must be utilized to handle the potential overflow of the eight element sum.

In addition this is the first routine to use anything other than the 16-bit Fixed Point number representation. The MXVALU is the Fixed Point integer resulting from the eight element average. The MXEXP is the TOTLSH (Total Shift) quantity which has been accumulated throughout in DTASCL (Data Scale). It is decremented by 1 to correct for the doubling which was inherent in the ORDER function (see Section 5.1.6.7).



ASP FFT Program  
FFT/FIT Program Description

The operations proceed as follows:

SCALE DATA:

Scale Data for Maximum Range

Call BDRYST for BDRYOK (Boundary OK) and FREELN (Free Length) values

Initialize Window Input and Window Output Pointers

Leave Max Pointer at First Value

Sum first Eight Values into Long Accumulator

Call SMBDCK (Smooth Boundary Check) to Test/Correct for Wraparound

SAVAVG: (Save Average)

Put Sum into Temporary Register, Divide by 8 for Average

Subtract Oldest Value from Total Sum

Save Averaged Sum at Oldest Location

If New Averaged Sum  $\Rightarrow$  Old Max Averaged Sum

Then New Max Sum = New Averaged Sum

Get Input Pointer to Next Magnitude

Call SMBDCK to Test/Correct for Address Wraparound

If Done,

Then Goto SMMXDN

Add New Magnitude into Long Accumulator

Goto SAVAVG

SMMXDN: (Smooth/Maximum Done)

Store Address of Maximum into MAXADDR

Compute Frequency Index of Maximum Value, Store as FMAX

Store Value at Maximum into MXVALU and MXEXP

Exit

SMBDCK: (Smooth Boundary Check)

If Input Pointer Past YSLWA+1  
Then Reset Past YSFWA

If Output Pointer Past YSLWA+1  
Then Reset Past YSFWA

Decrement Move Count by 4, If Done  
Then Set Done Flag True  
Else Set Done Flag False

Return via Reset to PC

### 5.2.3 CKFMAX (Check Maximum Value)

#### 5.2.3.1 Function

This routine does a number of things. They are:

- (1) Pass the number of Spectra Components to the Output List as LNFFT (32, 64, 128, or 256).
- (2) Correct the FMAX Fraction and Exponent Values by the Geophone Response Curve.
- (3) Correct the FMAX Values for the Geophone Power Gain.
- (4) Correct the FMAX Values for the Amplifier Power Gain.
- (5) Pass the Entries to the Output List.
- (6) Store the FMAX Index Number as F0 on the Output List. Zero GMAINT and GMAFRC (GAMMA Integer and GAMMA Fraction) on the Output List (these three entries are necessary in case of an aborted routine).
- (7) Assure a valid event. That is, assure:  
$$\text{Head Boundary} \leq \text{FMAX} \leq \text{Tail Boundary}$$
- (8) Return with Good or Bad Flag.

#### 5.2.3.2 Calling Program

CKFMAX is called by FFTMN.

#### 5.2.3.3 Inputs

The primary Input is the Frequency Magnitude Values in the FIFO, smoothed by the SMTGMX routine. Additional inputs are FLNPTR, with the FFT Length, the FIT Global Variables, and GAIN, the Amplifier Gain input by the Operator. The Instrument Correction Table is accessed for the FMAX index value. There are also some Internal Tables for the Head and Tail Boundaries, Correction Table Skip Lengths, and Spectra Lengths. Finally some Internal Constants for the

Geophone Gain Fraction and Exponent, and the Fix Table Normalize Fraction and Exponent are used.

#### 5.2.3.4 Internal Variables

There are no Internal Variables. Note: The FIT Global Variables are listed in the code as Internal. In reality since they are maintained by all code running afterward (including FFTMN) they are actually Global.

#### 5.2.3.5 Subroutines Used

SNMULT is used to multiply the energy values. In addition there are two Internal Subroutines. LNFI~~X~~ (Length Fix) is used to locate a pointer within the IN~~S~~FI~~X~~ table. FLSCCK (Full Scale Check) adjusts the MXNINT and MXNEXP values to full scale.

#### 5.2.3.6 Output

MXNINT and MXNEXP are the FMAX MXVALU and MXEXP values, corrected for the Geophone Response Curve, the Geophone Power Gain, and the Amplifier Power Gain. F0 receives FMAX. GMAINT (GAMMA Integer) and GMAFRC (GAMMA Fraction) are zeroed.

#### 5.2.3.7 Description

The routine proceeds as described in the Function Section 5.2.3.1.

ASP FFT Program  
FFT/FIT Program Description

The correction table is a table of 256 possible frequency entries from DC to 50 Hz to correct for the Geophone's Filtering Characteristics (run at 100 samples per second). It was generated by the program:

$$\text{INSFIX}(n) = \frac{\{[(2\pi \cdot 4.5)^2 + (2\pi \cdot n)^2]^2 + [2\pi \cdot 1.77]^2\}^{1/2}}{(2\pi \cdot n)^3}$$

The values are normalized for the  $n = 26$ th entry, corresponding to 5.0781 Hz. A non-zero-index normalization was required to maintain dynamic range as the rolloff becomes more severe. The frequency 5.0781 Hz was selected based upon geological considerations. Geophone and Amplifier Gain corrections were include to maintain ground motion to data output consistency.

For the Energy Magnitude Computation the value must be re-multiplied by  $n = 26$ 's actual magnitude. That is:

$$\begin{aligned} 3.40005E-2 &= \#0452 * 2^{(\#0000)} \\ &= \#45A2 * 2^{(\#FFFC)} \end{aligned}$$

As just mentioned there are geological considerations involved in not using events with maximum entries near DC or at the highest frequencies. Also, as will be seen in the GMACPT (Gamma Compute) program there are some mathematical reasons why the maximum can't be too far out. As it works out, this range of valid FMAX index values are:

```
for LNFFT = 32, 4 <= FMAX Index <= 13
for LNFFT = 64, 7 <= FMAX Index <= 33
for LNFFT = 128, 13 <= FMAX Index <= 81
for LNFFT = 256, 26 <= FMAX Index <= 153
```



It should be mentioned that consideration was also given to introducing the energy correction for spectra length necessary for harmonic analysis (see [Brig], pg. 137).

$$H(n/NT) = T/NT \text{ SUM}[h(kT)\exp(-j*2*\pi*n*k/N)] \text{ for } k = 0, 1, \dots, N-1$$

However this operation, dividing by delta T, is now done in the BOSS.

The routine proceeds as listed:

CKFMAX: (Check FMAX)

Store FFT Length (32, 64, 128, or 256) into FFTLN (FFT Length)

Prepare Pointer for Energy Correction

ENGCR: (Energy Correction)

Get Pointers to MXVALU and INSFIX Frequency Component  
Call LNFIX (Length Fix) for INSFIX Pointer

$MXNINT \leftarrow MXNINT * INSFIX(FMAX)$

NORMALIZE FOR 1ST INDEX VALUE:

$MXNINT \leftarrow MXNINT * FXNMFR$  (Fix Normalize Fraction)

$MXNEXP \leftarrow MXNEXP + FXNEXP$  (Fix Normalize Exponent)

Call FLSCCK (Full Scale Check) to test MXNINT and MXNEXP

CORRECT FOR GEOPHONE GAIN:

$MXNINT \leftarrow MXNINT * GPHFRC$  (Geophone Gain Fraction)

$MXNEXP \leftarrow MXNEXP + GPHEXP$  (Geophone Gain Exponent)

CORRECT FOR AMP GAIN:

$POWER \leftarrow \text{LOG}(\text{base } 2)\text{GAIN}$

$MXNEXP \leftarrow MXNEXP - POWER$

SET OUTPUT VARIABLES FOR POTENTIAL ABORTION:

Store FMAX into F0

Zero GMAINT and GMAFRC (GAMMA Integer and GAMMA Fraction)

ASP FFT Program  
FFT/FIT Program Description

TSTBDR:

If FMAX > Tail Boundary  
Then Exit with Error Flag (#00)

If FMAX < Head Boundary  
Then Exit with Error Flag (#00)

Exit with OK Flag (Not #00)

---

LNFIX: (Length Fix Internal Subroutine)

Get R(AC) to MXNINT (Maximum Energy Integer) INSFIX Value

Return via PC (not ready for another Call)

FLSCCK: (Full Scale Check Internal Subroutine)

Get R(AC) to MXNINT

INTCHK: (Integer Check)

If MXNINT 2nd MSB (Most Significant Bit) Equals "1"  
Then (since known positive) Goto EXFSCK

Shift MSNINT up by 1

Subtract 1 from MXNEXP

Goto INTCHK

EXFSCK: (Exit Full Scale Check)

Return via PC (not ready for another Call)

#### 5.2.4 GMACPT (GAMMA Compute)

##### 5.2.4.1 Function

GMACPT does a straightline fit for the slope of log(base 10) of the rolloff. The two points that are used for the line are four index averages around frequency values 6.25 and 18.75 Hz away from the FMAX index value (see Figure 5.2.4.1.a). That is:

$$\text{GAMMA} = \frac{(\log(\text{base10})Y1 - \log(\text{base10})Y2)}{(\log(\text{base10})X2 - \log(\text{base10})X1)} \quad (19)$$

Where: X1 = FMAX + 6.25 Hz  
X2 = FMAX + 18.75 Hz  
Y1 = AVGY(X1)  
Y2 = AVGY(X2)

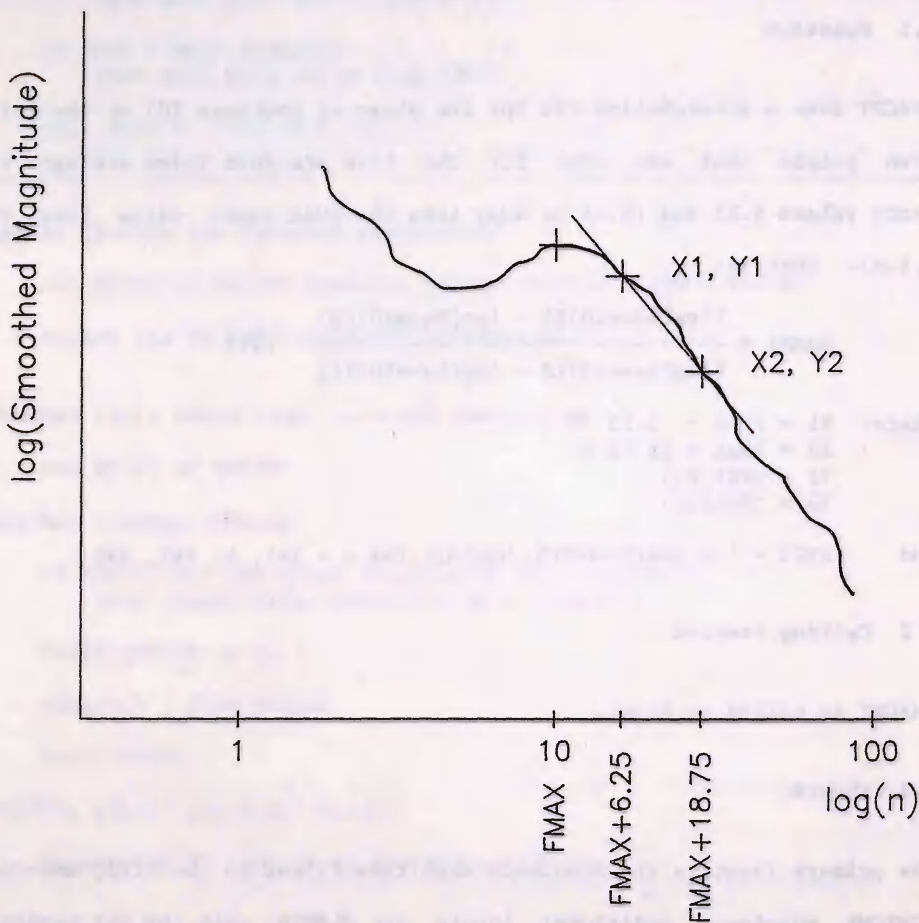
and AVGY = 1/4 SUM[SMOOTHED MAG(n)] for n = X-1, X, X+1, X+2

##### 5.2.4.2 Calling Program

GMACPT is called by FFTMN.

##### 5.2.4.3 Inputs

The primary Input is the Frequency Magnitude Values in the FIFO, smoothed by the SMTGMX routine. Additional inputs are FLNPTR, with the FFT Length, the LNFFT value, and the FIT Global Variables, especially FMAX and MXADDR. There are also some Internal Tables for X1, X2, Y1, and Y2 Offsets, Indexes, and Skip Values for the Spectra and INSFIX values. Finally there is a 127 entry Base 2 Log table.



$$GAMMA = \frac{\log(Y1) - \log(Y2)}{\log(X2) - \log(X1)}$$

Figure 5.2.4.1.a High Frequency Slope (GAMMA) Calculation

#### 5.2.4.4 Internal Variables

The following Internal Variables are used in the GMACPT routine.

Addr	Name	Description
30F4	NUMLOG	Numerator (Difference) Log 8-bit Mantissa, 16-bit Characteristic
30F7	DENLOG	Denominator (Difference) Log 8-bit Mantissa, 16-bit Characteristic
30FA	YXWORK	Working Y or X Temporary Scratchpad

Note: The FIT Global Variables are listed in the code as Internal. In reality since they are maintained by all code running afterward (including FFTMN) they are actually Global.

#### 5.2.4.5 Subroutines Used

SNMULT is used to multiply the energy values. LNFIX (Length Fix), an Internal Subroutine in CKFMAX, is used to locate a pointer within the INSFIX table. DIV, an RCA 32 bit by 16 bit divide, is also used. In addition there is an Internal Subroutine, LOG2, which calculates the base 2 log of a previously adjusted 7-bit number (see Description Section 5.2.4.7).

#### 5.2.4.6 Output

The only outputs are the GMAINT (GAMMA Integer) and GMAFRC (GAMMA Fraction) values.



5.2.4.7 Description

Computation of the logarithms will be accomplished base 2, utilizing a 127 entry look-up table. Equation (19) can be rewritten base 2 as:

$$\text{GAMMA} = (1/\log(\text{base}10)2) \frac{(\log(\text{base}2)Y1 - \log(\text{base}2)Y2)}{(\log(\text{base}2)X2 - \log(\text{base}2)X1)} \quad (20)$$

and  $\log(\text{base}10)2 = 0.3010$

Note: The conversion factor (0.3010) will be applied in the BOSS.

For all of the different length FFTs the frequency spectra spans from 0 to 50 Hz (100 samples per second). Thus the resolution for the FFT lengths 32, 64, 128, and 256 is 1.56, 0.78, 0.39, and 0.20, Hz respectively. As presented above the Y Average will be developed:

$$\text{AVGY} = 1/4 \text{ SUM}[\text{SMOOTHED MAG}(n)] \text{ for } n = X-1, X, X+1, X+2$$

Thus for the different resolutions the averaged (Hz) values will be:

FFT	Delta	4 Y1 Averaged Values					4 Y2 Averaged Values			
32	1.56	4.69	6.25	7.81	9.38		17.19	18.75	20.31	21.88
64	0.78	5.47	6.25	7.03	7.81		17.97	18.75	19.53	20.31
128	0.39	5.86	6.25	6.64	7.03		18.36	18.75	19.14	19.53
256	0.20	6.05	6.25	6.45	6.64		18.55	18.75	18.95	19.14

In all cases for the X value (n index value) the X1 point will be (FMAX + 6.25) Hz and X2 will be (FMAX + 18.75) Hz. Again for the different resolution for the various FFT lengths the Index Offset Lengths will be:

FFT	Delta	X1 Index Offset		X2 Index Offset	
32	1.56	4 (x 1.56 = 6.25)		12 (x 1.56 = 18.75)	
64	0.78	8 (x 0.78 = 6.25)		24 (x 0.78 = 18.75)	
128	0.39	16 (x 0.39 = 6.25)		48 (x 0.39 = 18.75)	
256	0.20	32 (x 0.20 = 6.25)		96 (x 0.20 = 18.75)	

This routine requires the computation of a logarithm. This will be accomplished base 2. Starting with the 16-bit fixed point number the characteristic is determined by the number of times it can be shifted up until the most significant bit becomes "1" (the more shifts the less the characteristic). The next highest seven bits of the resultant number are used to address a 127 element log table for the mantissa.

It will be seen that numeric representations other than 16-bit Fixed Point will be used by this routine. Integer-Exponent, Integer-Fraction, Characteristic-Mantissa, and 24-bit Fixed Point are all evident. They have been developed, and are utilized, as necessary.

The routine operates as listed:

GMACPT: (GAMMA Compute)

Get pointers to correct Length Table Y1 entry

Set Flag for Y1 pass

LYINIT: (Log2(Avg(Y)) Initialize)

Get pointers to value for Y, INSFIX, Skip values

Call LNFIX (Length Fix) for INSFIX pointer

Initialize Series Counter, 24-bit Accumulator, SNMULT pointer

MLTADD: (Multiply, then Add)

Check for Address Wraparound, If past YSLWA+1  
Then set past YSFWA

Accumulator  $\leftarrow$  Accumulator + (Y \* INSFIX)

Increment Y and INSFIX pointers

Decrement Series Counter, If not Done  
Then Goto MLTADD

Done with 4 Ys, Divide by 4 for Average

ASP FFT Program  
FFT/FIT Program Description

Call LOG2 for Log Base 2 of Averaged Y

If Y2 pass

Then Goto SBT2ND

Store LOG2(Y1) at 24-bit NUMLOG (Numerator Log)

Get pointers to Y2 entry

Set Flag for Y2 pass

Goto LYINIT

SBT2ND: (Subtract LOG2(AVG(Y2)))

Subtract Y2 Mantissa from Y1 Mantissa

If resultant Mantissa is Positive

Then Goto STNCHR

Add 1 to Y2 Characteristic (Subtracted Later), Add 1 to Mantissa

STNCHR: (Subtract Numerator Characteristic)

Subtract Y2 Characteristic from Y1 Characteristic

If Resultant Characteristic Negative (Error Condition)

Then Set Error Flag (#FFFF in GMAINT and GMAFRC)

Exit

DENCPT: (Denominator Computation)

Get Pointer to FMAX Index Value

Add X2 Index Offset

Call LOG2 for Log Base 2 of X2

Store LOG2(X2) in DENLOG (Denominator Logarithm)

Subtract (X2 to X1) Index Offset

Call LOG2 for Log Base 2 of X1

Subtract X1 Mantissa from X2 Mantissa

If resultant Mantissa is Positive

Then Goto STDCHR

Add 1 to X1 Characteristic (Subtracted Later), Add 1 to Mantissa

STDCHR: (Subtract Denominator Characteristic)

Subtract X1 Characteristic from X2 Characteristic (assured positive)

DIVIDE NUMERATOR CHAR/MAN BY DENOMINATOR CHAR/MAN:

CHRCCHK: (Characteristic Check)

If both Numerator and Denominator #00  
Then Goto DVNMDN

Shift Numerator and Denominator Char/Man down by 1

Goto CHRCCHK

DVNMDN: (Divide Numerator by Denominator)

Set Registers and Pointers for Divide

Call RCA 32-bit by 16-bit Divide

Store Quotient as GMAINT (GAMMA Integer)

With Remainder as Numerator, Same Denominator, Recall Divide

Save Quotient as GMAFRC (GAMMA Fraction)

Exit

---

LOG2: (Log(M(FRP)) Internal Subroutine)

If value equals #0000 (is assumed positive)  
Then Goto EXLOG2

Shift value until MSB = 1

Number of Shifts = Characteristic (more shifts = less characteristic)

Number now between "1" and "2"

Use next high 7 bits as address to 127 entry Log(base2) Table

EXLOG2: (Exit LOG2)

Return via PC

### 5.2.5 FOLPL (Compute F0 and Long Period Level)

#### 5.2.5.1 Function

This routine performs two functions. first it computes F0 by checking GAMMA.

```
If      GAMMA < 1.0, then ABORT with Flag
If 1.0 <= GAMMA < 2.5, then F0 = 2.06 * FMAX
If 1.5 <= GAMMA < 3.5, then F0 = 1.38 * FMAX
If 2.5 <= GAMMA < 4.5, then F0 = 1.27 * FMAX
If 3.5 <= GAMMA < 5.5, then F0 = 1.20 * FMAX
If 4.5 <= GAMMA < 6.5, then F0 = 1.16 * FMAX
If 5.5 <= GAMMA < 7.5, then F0 = 1.13 * FMAX
If 6.5 <= GAMMA < 7.5, then F0 = 1.10 * FMAX
If 7.5 <= GAMMA <      then F0 = 1.00 * FMAX
```

Then it computes the Long Period Level (LPL) by:

$$\text{LPL} = \frac{\text{Maximum Energy Value}}{0.9}$$
$$= 1.1 * \text{Maximum Energy Value}$$

Where the Maximum Energy Value (MXNINT and MXNEXP) is the energy at the FMAX value, already corrected for the Instrument Response and the Gain Factor.

#### 5.2.5.2 Calling Program

FOLPL is called by FFTMN.

#### 5.2.5.3 Inputs

Inputs are the FMAX value at the F0 location, the GAMMA value, and the MXNINT and MXNEXP values.



#### 5.2.5.4 Internal Variables

The FIT Global Variables are listed in the code as Internal. In reality since they are maintained by all code running afterward (including FFTMN) they are actually Global. NUMLOG and DENLOG are not used.

#### 5.2.5.5 Subroutines Used

This routine uses the SNMULT subroutine. Also internal to the routine is a GMACMP (GAMMA Compare) subroutine. This program checks the GAMMA value against a table entry, and returns a less than, or a greater than, flag.

#### 5.2.5.6 Output

If GAMMA is less than 1.0, then the routine is exited with the GAMMA as an out-of-bounds flag. Otherwise correct FO and Maximum Energy Values are returned.

#### 5.2.5.7 Description

The program operates as follows:

FOLPL: (FO and Long Period Level Compute)

Initialize pointers

If GAMMA < 1.00

Then exit with an Out-Of-Bounds GAMMA

If GAMMA => 1.50

Then Goto TBLTST

FO = 2.06 \* FMAX

(FMAX + FMAX + 0.6\*FMAX)

Goto LPLCPT

ASP FFT Program  
FFT/FIT Program Description

TBLTST: (Table Test)

Set Pointer to next Boundary Value  
(to 7.5, to 6.5, ..., to 1.5)

If GAMMA < Boundary  
Then Goto TBLTST

F0 = Factor \* FMAX

LPLCPT: (LPL Compute)

MXNINT = MXNINT + (0.10 \* MXNINT)

If Overflow  
Then MXNINT = MXNINT - 1  
MXNEXP = MXNEXP + 1

Exit

---

GMACMP: (GAMMA Compare Internal Subroutine)

Checks for GMAINT.GMAFRC < M(R(MQ)) + 0.50

Returns with DF = 0 If <, DF = 1 If =>

Compute GMAFRC - 0.50 (#4000)

If Result Negative  
Then Add 1 to M(R(MQ)) (subtracted next)

Compute GMAINT - M(R(MQ))

Return with Reset to PC, Ready for another Call, with DF Flag

### 5.3 Subroutines

The subroutines are organized for convenience into three separate files. The first file is called SUBROT and contains five separate routines (WRPTST, CS/SNMULT, ADDSTF, NEWVAL, and TRG/SHPSET). The second file contains the DTASCL routine. The third file, called SUBRT2 contains the BDRYST/CK routine. Each routine will be described individually.

#### 5.3.1 WRPTST (Wraparound Test)

##### 5.3.1.1 Function

For the operations where the addresses of the data being manipulated jump around this routine tests the R(MA) pointer to see if the computed address is beyond the end of the FIFO and reinitializes it if necessary. That is:

If  $(DIF = (R(MA) - (FIFO\ LWA+1))) \Rightarrow 0$

Then  $R(MA) = (FIFO\ FWA) + DIF$

The routine exits ready for another call.

##### 5.3.1.2 Calling Programs

This routine is used by the FFT, UNSCRM and ORDER programs.

##### 5.3.1.3 Inputs

It is assumed that R(ZAP1) points to the value for FIFO LWA+1, and the address of FIFO FWA is at the next location.

#### 5.3.1.4 Internal Variables

There are no Internal Variables. R(ZAP3.1) will be a flag in the subroutine.

#### 5.3.1.5 Subroutines Used

No (sub)subroutines are used.

#### 5.3.1.6 Output

R(MA) will be corrected if necessary.

#### 5.3.1.7 Description

RETURN:

Return ready for another Subroutine Call

WRPTST: (Wraparound Test Entry Point)

Subtract Boundary from Pointer

If Pointer less than Boundary  
Then Goto RSBDRY

Set Pointer past FWA

RSBDRY: (Reset Boundary)

Reset R(ZAP1) to Boundary Address

Goto RETURN

### 5.3.2 CSMULT and SNMULT (Cosine and Sine Multiply)

#### 5.3.2.1 Function

This program can be entered at two locations. CSMULT takes Operand B from M(CSVLU). SNMULT takes operand B from M(R(MQ)). In either case Operand A comes from M(R(MA)). The interrupts are disabled, the Multiplier Power is turned on, and the operands B and A are output to it. Control then returns to the calling program to do useful work while the hardware multiplier works (six long instructions). Upon return the result is shifted left by one to accommodate for the bit "lost" from the two two's-complement fractions multiplying. The result goes to M(R(AC)) and is also saved in R(ZAP3). The Multiplier Power is turned off, the interrupts reenabled, and the subroutine exited ready for another SNMULT call.

#### 5.3.2.2 Calling Programs

CSMULT is used by both the SHAPE and the FFT routines. SNMULT is used by the FFT, MAGAPX, CKFMAX, GMACPT and FOLPL routines.

#### 5.3.2.3 Inputs

Operand A, the data value, comes from M(R(MA)). For CSMULT operand B, the trig value, comes from M(CSVLU). For SNMULT operand B comes from M(R(MQ)).



ASP FFT Program  
FFT/FIT Program Description

5.3.2.4 Internal Variables

There are no Internal Variables. R(MA) and R(AC) (and R(MQ) if used) are left incremented by 2.

5.3.2.5 Subroutines Used

No (sub)subroutines are used.

5.3.2.6 Output

The result goes to M(R(AC)) and is also saved in R(ZAP3).

### 5.3.2.7 Description

#### CSMULT: (Cosine Multiply Entry Point)

Disable Interrupts, Turn On Multiplier Power

Output Cosine Value

Goto OUTMA

#### EXITM: (Exit Multiplier)

Turn Off Multiplier Power, Enable Interrupts

Return ready for another SNMULT Call

#### SNMULT: (Sine Multiply Entry Point)

Disable Interrupts, Turn On Multiplier Power

Get Sine Value, Output It

#### OUTMA: (Output M(R(MA)) Operand)

Output Operand A

Command to Multiply

Return to Calling Program while Hardware Works

#### MLTMDL: (Multiply Middle Return Point)

Upon Return, Get Result, Shift Up by 2, Store It

Exit through EXITM

### 5.3.3 ADDSTF (Add and Stuff)

#### 5.3.3.1 Function

This program computes LOREAL and LOIMAG or HIREAL and HIIMAG address values for the FFT routine. For Quad Address computation:

```
LOREAL = BASE + COLPTR  
LOIMAG = BASE + COLPTR + 2  
HIREAL = BASE + COLPTR + COLBIT  
HIIMAG = BASE + COLPTR + COLBIT + 2
```

For the UNSCRM routine, the Quad Addresses are:

```
NMREAL = BASE + NMPTR  
NMIMAG = BASE + NMPTR + 2  
SWREAL = BASE + SWPTR  
SWIMAG = BASE + SWPTR + 2
```

In each case the locations of the Quad Address pointers are sequential (separated by 2) and are computed a pair at a time. Functionally, for the registers initialized prior to entry, the operations are:

```
M(R(ZAP1)) = R(MP) + M(R(FRP))  
M(R(ZAP1)+2) = R(MP) + M(R(FRP)+2)
```

The routine exits ready for another subroutine call.

#### 5.3.3.2 Calling Programs

ADDSTF is called by FFT and UNSCRM.

### 5.3.3.3 Inputs

The Input Values are in the R(ZAP1) and R(MP) registers, and in M(R(FRP)) and M(R(FRP))+2.

### 5.3.3.4 Internal Variables

There are no Internal Variables. R(ZAP2) is used as a temporary storage location.

### 5.3.3.5 Subroutines Used

No (sub)subroutines are used.

### 5.3.3.6 Output

R(ZAP2) is returned updated by M(R(FRP)).

### 5.3.3.7 Description

RETURN:

Return ready for another Subroutine Call

ADDSTF: (Add and Stuff Entry Point)

$M(R(ZAP1)) = R(MP) + M(R(FRP))$

Store to --REAL

Increment Value by 2

Store to --IMAG

Exit through RETURN

ASP FFT Program  
FFT/FIT Program Description

### 5.3.4 NEWVAL (New Value)

#### 5.3.4.1 Function

This routine computes and stores the new entries for the quad of data for the FFT routine, a pair at a time. That is, either:

$$\begin{aligned} M(\text{LOREAL}) &= M(\text{LOREAL}) + \text{TREAL, or} \\ M(\text{HIREAL}) &= M(\text{HIREAL}) - \text{TREAL.} \end{aligned}$$

Or:

$$\begin{aligned} M(\text{LOIMAG}) &= M(\text{LOIMAG}) + \text{TIMAG, or} \\ M(\text{HIIMAG}) &= M(\text{HIIMAG}) - \text{TIMAG.} \end{aligned}$$

Address pointers LOREAL, LOIMAG, HIREAL, and HIIMAG are located sequentially in memory, 2 bytes apart. Thus for preinitialized pointers:

R(FRP) points to LOREAL or LOIMAG, and  
R(ZAP1) points to TREAL or TIMAG

This is accomplished by:

$$\begin{aligned} M(M(R(\text{FRP}))) &= M(M(R(\text{FRP}))) + M(R(\text{ZAP1})) \\ M(M(R(\text{FRP})+4)) &= M(M(R(\text{FRP})+4)) - M(R(\text{ZAP1})) \end{aligned}$$

This routine exits ready for another subroutine call.

#### 5.3.4.2 Calling Programs

NEWVAL is called by FFT.

#### 5.3.4.3 Inputs

R(FRP), M(R(FRP)), M(M(R(FRP))), M(M(R(FRP))+4), and M(R(ZAP1)).



#### 5.3.4.4 Internal Variables

There are no Internal Variables.

#### 5.3.4.5 Subroutines Used

No (sub)subroutines are used.

#### 5.3.4.6 Output

The new column's data is computed for the correct pointer locations.

#### 5.3.4.7 Description

RETURN:

Return ready for another Subroutine Call

NEWVAL: (New Value Entry Point)

Load the Updating Value (@M(R(ZAP1)))

Subtract from the Value to be Updated (@M(M(R(FRP))))

Add to that Value and Store at HI----

Store Subtracted Value at LO----

Exit through RETURN

### 5.3.5 TRGSET and SHPSET (Trig and Shape Set)

#### 5.3.5.1 Function

These subroutines compute the address of the pointers for the 256 entry trig table. That is:

$$\text{TRGPTR} = \text{BIT REVERSAL}(\text{TRGCTR})$$

Bit reversal is a well known algorithm necessary for a couple of operations in the FFT algorithm. Where the initial bit pattern is:

$$X_n, X_{n-1}, \dots, X_1, X_0$$

The pattern after bit reversal is:

$$X_0, X_1, \dots, X_{n-1}, X_n$$

Or, for example, for length 8:

000	results in	000
001	results in	100
010	results in	010
011	results in	110
100	results in	001
101	results in	101
110	results in	011
111	results in	111

The pointers are generated:

$$\begin{aligned}\text{SNPTR} &= \text{TRGFWA} + \text{TRGPTR} \\ \text{CSPTR} &= \text{TRGLWA} - \text{TRGPTR}\end{aligned}$$

Only the first quarter of the sine wave has been included, from  $\sin(0) = 0$  to  $\sin(\pi/2) = 1$  (or #7FFF). Since trig values are needed for sine and cosine from 0 to  $\pi$ , corrections must be included. Sine values will be figured by a pointer added to the base address ( $\text{SNPTR} = \text{TRGFWA} + \text{TRGPTR}$ ). If SNPTR is past TRGLWA then the routine returns it back as many as it was off the end. Cosine values will be computed by a pointer subtracted from the last address ( $\text{CSPTR} =$

TRGLWA - TRGPTR). If the address is after TRGFWA (0 to  $\pi/2$ ) it stores the entry as is at CSVLU. If the pointer is in front of TRGFWA (between  $\pi/2$  and  $\pi$ ). it returns back as many entries as it was in front and stores that value, negated, into CSVLU.

There is a second entry point used by the SHAPE routine, at SHPSET (Shape Set), which uses the CSVLU computing algorithm.

The subroutine exits ready for another TRGSET call.

#### 5.3.5.2 Calling Programs

These subroutines are used by the FFT and SHAPE routines, respectively.

#### 5.3.5.3 Inputs

R(MP) must have the TRGCTR value.

#### 5.3.5.4 Internal Variables

The following variables are used in the routine, and returned to the calling function.

Addr	Name	Description
30EA	TRGCTR	Trig Counter
30EC	CSPTR	Cosine Pointer
30EE	SNPTR	Sine Pointer
30FO	TRGPTR	Trig Pointer

ASP FFT Program  
FFT/FIT Program Description

5.3.5.5 Subroutines Used

No (sub)subroutines are used.

5.3.5.6 Output

The variables listed above, TRGCTR, CSPTR, SNPTR, and TRGPTR are output. In addition the CSPTR value is stored in CSVLU.

5.3.5.7 Description

RETURN:

Return ready for another TRGSET Subroutine Call

TRGSET: (Trig Set Entry Point)

Shift TRGCTR into TRGPTR 10 Times

Compute  $SNPTR = TRGFWA + TRGPTR$

If  $SNPTR \leq TRGLWA$   
Then Goto CSCMPT

Return SNPTR back into Trig Table

CSCMPT: (Cosine Compute)

Compute  $CSPTR = TRGLWA - TRGPTR$

SHPSET: (Shape Set Entry Point)

If  $CSPTR \Rightarrow TRGFWA$   
Then Goto CSVLST

Return CSPTR back into Trig Table

Negate Cosine Value

CSVLST: (Cosine Value Store)

Store into CSVLU

Exit through RETURN

### 5.3.6 DTASCL (Data Scale)

#### 5.3.6.1 Function

DTASCL scales the data up for maximum dynamic resolution, yet assures that there are no overflow errors. For a given input MAXSHIFT it scales the data so that the greatest magnitude is between +MAXVALUE and -MAXVALUE, where:

MAX- SHIFT	Maximum Positive Number					Maximum Negative Number				
	Hex	Binary				Hex	Binary			
7	#7FFF	0111	1111	1111	1111	#8000	1000	0000	0000	0000
6	#3FFF	0011	1111	1111	1111	#C000	1100	0000	0000	0000
5	#1FFF	0001	1111	1111	1111	#E000	1110	0000	0000	0000
4	#0FFF	0000	1111	1111	1111	#F000	1111	0000	0000	0000
3	#07FF	0000	0111	1111	1111	#F800	1111	1000	0000	0000
2	#03FF	0000	0011	1111	1111	#FC00	1111	1100	0000	0000
1	#01FF	0000	0001	1111	1111	#FE00	1111	1110	0000	0000
0	#00FF	0000	0000	1111	1111	#FF00	1111	1111	0000	0000

The routine incorporates internal Boundary Set and Boundary Check code. Then the FIFO is scanned from the start for the length to find the highest significant place, or until a number occurs for which the MSB is active. This is done by checking the words high byte with a 1's AND mask for positive entries and a 0's OR mask for negative entries. This maximum magnitude is then compared with the desired MAXVALUE to see if shifting is required. If shifting is required the TOTLSH (Total Shift) entry is updated, then the FIFO is shifted, either up by one, up by more than one, down by one, or down by more than one (four routines were included for increased speed for this commonly run program). It should be noted that for shifting down, care must be taken to assure correct sign extensions for both positive and negative numbers. The routine exits ready for another subroutine call.



5.3.6.2 Calling Programs

DTASCL is called by SCALE, FFT, ORDER, MAGAPX, and SMTGMX.

5.3.6.3 Inputs

It is assumed that the Start Pointer is in R(MP), the Length Pointer is at M(R(FRP)), the MAXSHIFT value is in R(ZAP2), the End Boundary address is in R(ZAP1) and the Start Boundary address is at M(R(ZAP1)+2)

5.3.6.4 Internal Variables

The following variables are used internally.

Addr	Name	Description
3E01	THBDFL	This Boundary Flag
3E03	BDOKRM	Boundary OK Remember
3E05	LNFREE	Length Free
3E07	UNDRSH	Under Shift Value
3E09	SCSTRT	Scale Start Address
3E0B	SCLN	Scale Length Count

5.3.6.5 Subroutines Used

No (sub)subroutines are used.

5.3.6.6 Output

One output is the data in the FIFO, correctly shifted. Also the TOTLSH value is updated if necessary.

### 5.3.6.7 Description

#### RETURN:

Return ready for another Subroutine Call

#### DTASCL: (Data Scale Entry Point)

Save Start Address, Length Count, MAXSHIFT

Compute FREELN, Store

Test for Wraparound Condition, If OK  
Then Set BDOKRM True  
Else Set BDOKRM False

Set Maximum Shift Count to 7

Load AND Mask (#FF) and OR Mask (#00)

#### MEMTST: (Memory Test)

Test Memory Entry for Shift OK, If Too Many  
then Goto SHFTLS

#### NXTCHK: (Next Memory Check)

If Done Testing  
Goto NOWSCL

If Boundary OK Flag True  
Goto MEMTST

Decrement Free Length Counter, If Not LWA+1  
Goto MEMTST

Set Pointer to FWA, Set Boundary OK Flag True, Goto MEMTST

#### SHFTLS: (Shift Less)

Decrement MAXSHIFT, If at Limit (#00)  
Then Goto NOWSCL

Adjust AND and OR Mask, Goto MEMTST

ASP FFT Program  
FFT/FIT Program Description

NOWSCL: (Now Scale)

Compute Shift Value, If #00  
Then Exit Through RETURN

Adjust TOTLSH Value

Retrieve Start Address, Length Count, MAXSHIFT, BDOKRM

If Shift Value Negative  
Then Goto SHDOWN

If Shift > 1  
Then Goto SHUPMR

SHUPON: (Shift Up by 1)

Shift Value Up by 1

Decrement Length Count, If Done  
Then Exit through RETURN

If BDOKFL True  
Then Goto next SHUPON

Decrement LNFREE, If #00  
Then Reset Pointer to FWA, Set BDOKFL True

Goto next SHUPON

SHUPMR: (Shift Up by More Than 1)

Get word, Shift Up Number of Times, Store Back

Decrement Length Count, If Done  
Then Exit through RETURN

If BDOKFL True  
Then Goto next SHUPMR

Decrement LNFREE, If #00  
Then Reset Pointer to FWA, Set BDOKFL True

Goto next SHUPMR

SHDOWN: (Shift Down)

If Shift < -1  
Then Goto SHDNMR

SHDNON: (Shift Down By 1)

Shift Value Down by 1, With Sign Extend

Decrement Length Count, If Done  
Then Exit through RETURN

If BDOKFL True  
Then Goto next SHDNON

Decrement LNFREE, If #00  
Then Reset Pointer to FWA, Set BDOKFL True

Goto next SHDNON

SHDNMR: (Shift Down By More Than 1)

Get word, Shift Number of Times With Sign Extend, Store Back

Decrement Length Count, If Done  
Then Exit through RETURN

If BDOKFL True  
Then Goto next SHDNMR

Decrement LNFREE, If #00  
Then Reset Pointer to FWA, Set BDOKFL True

Goto next SHDNMR

### 5.3.7 BDRYST/CK (Boundary Set and Check)

#### 5.3.7.1 Function

These subroutines are used by sequential functions to check the length of the operation against the length of the FIFO for a potential wraparound condition. There are two entry points to this routine. The first computes the length from the First Address to LWA+1, that is the Free Length, and compares it to the Move Length for possible Boundary OK condition. It sets BDRYOK, FREELN, FWA, and MOVECT. The second entry point decrements FREELN and checks for LWA+1 (if required). If necessary it resets R(MA) to FWA. It then decrements MOVECT to check for Done. If Done it returns with D = 00; else D <> 00.

Both routines exit ready for a BDRYCK call.

#### 5.3.7.2 Calling Programs

BDRYST/CK are used by the SHAPE and MAGAPX programs while BDRYST alone is called by SMTGMX (which has its own boundary check program, SMBDCK).

#### 5.3.7.3 Inputs

BDRYST requires that R(MA) have the Start Address (it is left unchanged), and R(MP) must point to Move Count (it is incremented by 2). R(ZAP1) must point to LWA+1 and M(R(ZAP1)+2) must contain the FWA. BDRYCK utilizes R(MA) the Address Pointer, and FREELN.



#### 5.3.7.4 Internal Variables

There are no internal variables. R(ZAP1), R(FRP), and R(ZAP2) are used in BDRYST. R(MAP1) and R(ZAP2) are used in BDRYCK.

#### 5.3.7.5 Subroutines Used

No (sub)subroutines are used.

#### 5.3.7.6 Output

The values given above - BDRYOK, FREELN, FWA, and MOVECT are set by BDRYST. BDRYCK decrements FREELN and resets R(MA) if necessary. The D accumulator is used as a done flag.

#### 5.3.7.7 Description

BDRYST: (Boundary Set Entry Point)

Compute  $FREELN = (LWA+1) - \text{Start Address}$

If  $MOVECT \leq FREELN$   
Then BDRYOK = True  
Else BDRYOK = False

EXBDSC: (Exit Boundary Set/Check)

Return ready for another BDRYCK Subroutine Call

BDRYCK: (Boundary Check Entry Point)

If BDRYOK is True  
Then Goto CKMVDN

Decrement FREELN, If not #0000  
Then Goto CKMVDN

Reset Pointer to FWA, Set BDRYOK True

ASP FFT Program  
FFT/FIT Program Description

CKMVDN: (Check Move Done)

Decrement MOVECT, If #0000  
Then Set Done Flag #00  
Else Done Flag is <> #00

Exit through EXBDSC

## 6.0 RUN TIME PROGRAMS

As was mentioned in Section 3.5, Development System Support, no debug support was provided. Thus for setting breakpoints, testing or setting registers and memory locations, or for restarting the program, Run Time code had to be developed. Three types were developed: (1) a System Run (SYSRUN) program; (2) a FIFO loading (LDFIFO) program; and (3) a series of Tracing programs. Additionally a Sine Wave table (SINWAV) was used to load the FIFOs with various frequency sine waves, and PQUAKE and SQUAKE earthquake listings were captured for convenient regeneration of time series data.

These programs are interfaced to the Program Code with manually inserted Long Branches. These Long Branches were originally overwritten onto the code; later No-Operations (NOPs, #C4C4C4 for the 1802) were inserted at strategic locations for the breakpoints. In this way the FFT/FIT subprograms were cut up into manageable (debuggable) sections.

### 6.1 Operation

#### 6.1.1 Purpose

The general purposes of the Run Time Code are:

- (1) SYSRUN
  - Saves RCA Development System (Monitor) Registers,
  - Loads variable and static FFT/FIT Program Registers,
  - Calls the FFT/FIT Program to be run,
  - upon FFT/FIT Program Return, saves Program Registers,
  - may Print Registers and/or Memory data, or nothing,
  - Returns to Monitor or to FFT/FIT Program.

ASP FFT Program  
Run Time Programs

(2) LDFIFO Loads either P or S FIFO with either:

- Zeros;
- User directed Geometric Series; or
- Sine Wave of selected frequency.

(3) Trace Programs These are overlaid onto the FFT/FIT Program, SYSRUN, and LDFIFO code to set desired Control Flow for the test in progress. Control Flow will usually be:

- Call LDFIFO to input desired data (the Trace Program has inserted the required LDFIFO Table entries);
- Branch to SYSRUN to Save System, then Load FFT/FIT Program Registers;
- SYSRUN then Calls the FFT/FIT Program;
- the Trace Program has set the desired Breakpoints and, upon Exit, SYSRUN saves the FFT/FIT Program Registers;
- the Trace Program has filled the required Table entries for desired Printouts;
- after Printouts the FFT/FIT Program is either Recalled, or Control is returned to the Monitor.

#### 6.1.2 Data Structures

The Memory Locations, Routine Locations, Variables, and Tables most used by the Run Time Code are listed here for convenience.

##### 6.1.2.1 Memory Locations

0000 - 3FFF      Memory

0000 - 2FFF      ROM

1700 - 2637	FFT Programs (Set #1)
2660 - 26F6	FFT Constants (SYSSET)
26FE - 27FF	TRGTBL (Trig Table)
2800 - 2BC1	FFT Subroutines
2C00 - 2CD8	FFT Programs (Set #2)

3000 - 3FFF      RAM

3000 - 30FF	FRAM (Fast RAM)
3000 - 30C3	WORKER Variables
30C4 - 30DF	FFT Global Variables
30E0 - 30FF	FFT Local Variables
30E0 - 30FA	FIT Global Variables
3601 - 3A00	YPFIFO
3A01 - 3E00	YSFIFO

The Run Time Programs are loaded into the Development System memory at locations above the WORKER locations (above #4000, to #7FFF).

### 6.1.2.2 Routine Locations

#### WORKER Routines:

Addr	Name	Description
0033	CALL	Address of Call Routine
0043	RETN	Address of Return Routine
0050	INTP	Address of Interrupt Routine

#### FFT Routines:

Addr	Name	Description
1700	FFTMN	FFT Main
1700	P-FFT	P-FFT Call
1703	S-FFT	S-FFT Call
176D		SFFT EXIT, after Transmitting S-DATA to BOSS, right before EXITING back to Calling routine (used in 8085 FULL TRACE as SFFT Break)
1796		PFFT EXIT, after Transmitting P-DATA to BOSS, right before EXITING back to Calling routine (used in 8085 FULL TRACE as PFFT Break)
17A0	LNSET	Length Set
17A0	PLNSET	P-Length Set
17A3	SLNSET	S-Length Set
18A0	SCALE	Scale
18A0	PSCALE	P-Scale
18A3	SSCALE	S-Scale
1900	SHAPE	Shape
1900	PSHAPE	P-Shape
1903	SSHAPE	S-Shape
1A60	FFT	FFT Routine
1C00	UNSCRM	Unscramble
1D20	ORDER	Order

#### FIT Routines

Addr	Name	Description
2000	MAGAPX	Magnitude Approximate
2100	SMTGMX	Smooth and Tag Maximum
2250	CKFMAX	Check F0 Maximum
23F0	GMACPT	GAMMA Compute
2C00	FOLPL	FOLPL Compute



ASP FFT Program  
Run Time Programs

Run Time Routines

SYSRUN Functions

Addr	Name	Description
7000	SYSRUN	System Run (to #7400)
7003	RGPRNT	Register Print, then Quit
7006	PRGDUN	Program Done, so Quit
7009	RGDATA	Print Register Data
700C	FFTRAM	Print FFT RAM
700F	PFTDTA	Print P-FFT Data
7012	SFTDTA	Print S-FFT Data
7015	FFTDTA	Print FFT Data
701A	SYSRUN	(after LBR and DIS ,#00)
706D		In SYSRUN, where FFT/FIT Program Called from
70EE		Directory to Print Outgoing Register Table
712F		Directory to Print FFT RAM (#30A0 + #60)
7137		Directory to Print FFT OUPUT (YSFFT Data)
713D	PRMEM	Print Memory
725E		Ingoing Register Table
73E0		Outgoing Register Table

LDFIFO Functions

Addr	Name	Description
5100	LDFIFO	Load FIFO (to #5244)
5100	YPZERO	Zero YPFIFO
5103	YSZERO	Zero YSFIFO
5106	MANINP	Manual Input
5152		Start of MANINP
51D2		End of MANINP
522A		MANINP Input Table
522A		MANINP Input P-Table
522E		MANINP Input S-Table
5232		MANINP Value Table
5109	MODSIN	Modified Sine Input
51D5		Start of MODSIN
521B		End of MODSIN
521E		MODSIN Input Table
521E		MODSIN Input P-Table
5222		MODSIN Input S-Table
5226		MODSIN FIFO Pointer and Skip Value

Development System Monitor Programs

Addr	Name	Description
80EF	DELAY1	Delay for Print Operation
819C	TYPE5D	Print an ASCII Character (CR, LF, SP)
81AE	TYPE2	Print a Byte as 2 Hex Characters
83F9	GOUT20	Goto UT20 - Return to Monitor (branches to #8281 ENTER)
8C00	WRAM	Registers Stored in RAM

### 6.1.2.3 Variables

#### WORKER Variables

Addr	Name	Description
-----	-----	-----
3046	SLAST	Time for the Last YSFIFO
3048	DT	ST Minus PT

#### FFT Global Variables

Addr	Name	Description
-----	-----	-----
30C4	PFTFLG	P-FFT or S-FFT Flag
30C5	BDYOK	Boundary OK Flag
30C6	FREELN	Free Length Count
30C8	FWA	First Word Address of Move Out FIFO
30CA	MOVECT	Move Count
30CC	PFFTLN	P-FFT Length
30CE	SFFTLN	S-FFT Length
30D0	FFTLN	FFT Length
30D2	PLNPTR	P-FFT Length Pointer
30D4	SLNPTR	S-FFT Length Pointer
30D6	FLNPTR	FFT Length Pointer
30D8	BASE	Start Address for FFT Data
30DA	TOTLSH	Total Shift Value
30DC	PSHIFT	P-FIFO Shift Value
30DE	CSVLU	Cosine Value Temporary Storage

#### FIT Global Variables

Addr	Name	Description
-----	-----	-----
30E0	MXADDR	Address of Maximum
30E2	FMAX	Frequency Index of Maximum
30E4	MXVALU	Maximum Value (Integer)
30E6	MXEXP	Maximum Value (Exponent "Shift" Value)
30E8	LNFFT	Length of FFT
30EA	F0	F0, Corner Frequency
30EC	GMAINT	GAMMA Integer
30EE	GMAFRC	GAMMA Fraction
30F0	MXNINT	Maximum Energy Integer
30F2	MXNEXP	Maximum Energy Exponent

ASP FFT Program  
Run Time Programs

6.1.2.4 Tables

Run Time Tables

Addr	Name	Description
4000	SINWAV	Sine Wave (to #41FF)
7A00	PQUAKE	P-Quake Data (to #7A7F)
7B00	SQUAKE	S-Quake Data (to #7BFF)

SINWAV is listed in Appendix B.2.1. PQUAKE and SQUAKE are not included in this report.

## 6.2 SYSRUN (System Run)

### 6.2.1 Function

This program has five main sections:

- (1) Begin SYSRUN, Save System Registers, Load Program Registers, Call FFT/FIT Program;
- (2) Upon FFT/FIT Return, Save Program Registers, (maybe Print them), Restore System Registers, Return to Monitor;
- (3) Directories for Printing Memory Blocks;
- (4) Code for Printing Memory Blocks;
- (5) Register Storage Locations.

Utilizing this code, with the aid of Trace Routines, the user may either:

- (1) Break from the User Program  
Save User Registers  
Return System Registers  
Print User Registers  
Return to Monitor, or
- (2) Break from the User Program  
Save User Registers  
Return System Registers  
Return to Monitor, or
- (3) Break from the User Program  
Save User Registers  
Return System Registers  
Print User Registers  
Return User Registers  
Return to User Program

### 6.2.2 Calling Program

The calling program varies, but is usually directly from the Monitor program (via User direction) or after the LDFIFO routine.

### 6.2.3 Inputs

The User must fill the Print Directories with the address locations which are to be printed.

### 6.2.4 Internal Variables

The Register Files could be considered to be Internal Variables.

### 6.2.5 Subroutines Used

The Development System Routines listed in the Section 6.1.2.2 are called. Also the program has an Internal Subroutine, Print Memory (PRMEM).

### 6.2.6 Output

Registers and Memory Blocks may be printed.



### 6.2.7 Description

#### SYSRUN: (System Run)

Save System Registers

Load Variable and Static FFT/FIT Program Registers

Branch to FFT/FIT Program Code

---

#### RGPRNT: (Register Print)

Print Registers, then Quit

#### PRGDUN: (Program Done)

Save Registers

Return to System Monitor

---

#### Entry Points for Printing Memory Blocks

#### RGDATA: (Print Register Data)

#### FFTRAM: (Print FFT RAM Data)

#### PFTDTA: (Print P-FFT Input Data)

#### SFTDTA: (Print S-FFT Input Data)

#### FFTDTA: (Print FFT Output Data)

---

#### PRMEM: (Print Memory Internal Subroutine)

Called to Print a Block of Memory, it handles FIFO Address Wraparound  
R6 is a Pointer to Directory of:

Start Address,

Last Address,

First Address.

R6 + 2 is a Pointer to the Length to be Printed.

Save Program Registers

Get Addresses, Values at Pointers

Prepare for Printout

Type CR/LF

ASP FFT Program  
Run Time Programs

LNOUT: (Line Out)

Type CR/LF

Print Data Address

TSPACE: (Type Space)

Type a Space

TLOOP: (Type Word Loop)

Type one Byte

If Done with Block  
Then Goto EXPRMM

If Address Wraparound  
Then Reset to FWA

If End of Line  
Then Goto LNOUT

If End of Data Word  
Then Goto TSPACE  
Else Goto TLOOP

EXPRMM: (Exit Print Memory)

Restore Program Registers

Return to Calling Program

---

TABLES:

INTBL: (Ingoing Register Table)

OUTBL: (Outgoing Register Table)

### 6.3 LDFIFO (Load FIFO)

#### 6.3.1 Function

This routine inputs data to the selected FIFO. Either the FIFO is zeroed, loaded with a geometric function (such as a square or triangular wave), or loaded with a sine wave of a selected frequency. There are four self-contained programs in this routine:

5100	YPZERO	Zero YPFIFO
5103	YSZERO	Zero YSFIFO
5106	MANINP	Manual Input
5109	MODSIN	Modified Sine Input

#### 6.3.2 Calling Program

The calling program is usually the Monitor program, under User direction.

#### 6.3.3 Inputs

The selections for the MANINP and MODSIN generating tables must be input.

#### 6.3.4 Internal Variables

There are no Internal Variables.

#### 6.3.5 Subroutines Used

The System Monitor GOUT20 (return to Monitor) routine is called to Exit.

### 6.3.6 Output

The FIFOs are filled with the desired data.

### 6.3.7 Description

YPZERO: (Zero YPFIFO)

Initialize Pointer to YPLWA

PTZROP: (Put Zero into YPFIFO)

Store #00

Decrement Pointer, If Not Done  
Then Goto PTZROP

Exit

---

YSZERO: (Zero YSFIFO)

Initialize Pointer to YSLWA

PTZROS: (Put Zero into YSFIFO)

Store #00

Decrement Pointer, If Not Done  
Then Goto PTZROS

Exit

---

MANINP: (Manual Input)

TABLE1 must have:

Start Address;

LWA+1, FWA Pointer Location;

Initial Value;

Length of Block;

Increment/Decrement Value.

Note: May be additional Blocks

Enter #ABxx to Stop

Initialize Pointers to Addresses, Lengths, Values

INFRLN: (In Free Length)

Get Boundary, subtract Start Address (= Free Length)

Check for Free Length, If Boundary Good  
Then Store Boundary OK Flag True  
Else Store Boundary OK Flag False

Load Block Length

STRVLU: (Store Value)

Store Value

Increment/Decrement Value by Delta

Check for Address Wraparound, If LWA+1  
Then Set to FWA

Decrement Block Length, If Not Done  
Then Goto STRVLU

Check for Not Done, If Not #AB  
Then Goto INFRLN

Exit

---

MODSIN: (Modified Sine Wave)

TABLE2 must have:

FWA, LWA+1 Pointer Location;  
Skip Value.

Initialize Pointers to FIFO, Sine Table

PASS: (Pass Sine Value)

Pass Sine Value

If FIFO Pointer to LWA  
Then Exit

Increment Sine Pointer, If Past LWA  
Then set into FWA

Goto PASS



#### 6.4 Data Input

LDFIFO, a program to load the FIFO with either zeros, a geometric sequence, or a sine wave of selected frequency, has already been explained (Section 6.3). Three other capabilities exist for loading the FIFO with data.

- (1) analog magnetic tape recordings of various earthquake data.
- (2) a simulated earthquake generated by a dedicated 8085 microcomputer system which outputs a frequency wave at a given amplitude to establish a steady-state baseline, then a P-Wave, and finally a S-Wave. This capability was supplied by the technicians at the UCB Seismographic Station and proven to be invaluable to the system debugging and validation.
- (3) the stored, digitized recordings of the 8085 earthquake sequence.

It should be noted that the third method tests the FFT/FIT operation only, while the first two capabilities test the entire WORKER, and even the BOSS and WORKER, system.

## 6.5 Trace Programs

As has previously been mentioned, Trace Programs were developed to direct the Control Flow of the procedures while testing the operation of the FFT/FIT code. As such they were short, terse patches that were written on an "as needed" basis. Numerous versions of the programs were written, and they proved essential in supporting the Debugging procedure.

Five samples of these code sections will be presented: (1) FFT Complete Trace; (2) FFT Out Trace; (3) FFT No Trace; (4) Multi FFT; and (5) 8085 Full Trace. The code itself is contained in Appendix B. These programs will not be described in any great detail but are included to give the reader an example of this method and capability. In addition their operation will explain how the Program Validation Code listed in Appendix D was generated.

### 6.5.1 FFT Complete Trace

FFT Complete Trace is the type of program used to generate the listings contained in the Program Validation Output, Subprogram Validation Series (Appendix Section D.1). With this program after either MANINP or MODSIN is run SYSRUN is called, branching either to PFFT or SFFT. Then as the program continues, at the end of every significant section (at the end of every subprogram, and often at locations internal to the subprograms) Print Memory (PRMEM) is called to print the outputs of that section. At the conclusion of the entire FFT/FIT call control is returned to the System Monitor (@ line #0079).

#### 6.5.2 FFT Out Trace

FFT Out Trace is similar to FFT Complete Trace except that only the final outputs of the FFT Call are printed. This is the type of code which generated the Program Validation Input/Output results listed in Appendix Section D.2 except of course for the lack of an Input print. It should also be noted that this code may be overlaid onto the FFT Complete Trace to change the desired output.

#### 6.5.3 FFT No Trace

This code is used to overlay onto either of the two previous programs to turn off the output printings.

#### 6.5.4 Multi FFT

Multi FFT is used to test the FFT/FIT function for sensitivity to the starting address of the time series data (the Address Wraparound condition). This is accomplished by passing either the PQUAKE or SQUAKE data to the YP or YSFIFO, initially at the first address, then incremented by two, up to the last address. FFT/FIT is called each time, and at each return the FIT variables are printed.

#### 6.5.5 8085 Full Trace

This program is used in conjunction with the 8085 Earthquake Generator (Section 6.4). It supports normal run time operation for the whole WORKER system, yet breaks to print the FIT output variables. In addition, if out-of-bounds data results (for PFFT F0  $\leq$  5, for SFFT F0  $\leq$  6) the program also breaks to print the entire extent of FFT data. This was necessary to trap an intermittent error resulting from a difference in technique between checking for

intermittent error resulting from a difference in technique between checking for the address wraparound condition between the WORKER and the FFT/FIT code (see Section 6.6).

## 6.6 Problems

Two significant problems occurred at the interface between the WORKER and the FFT/FIT programs. The first problem involves the use of the SHOLD value with the P-FFTs. SHOLD is the value that allows or prevents the WORKER from logging seismographic entries into the YSFIFO. For the P-FFT the YSFIFO is used for the in-place computation and thus SHOLD should be set to prevent data overwrite. This was not done in the WORKER and must be set, and later cleared, by the FFTMN program (Section 5.0.1).

The second problem involves checking for the Address Wraparound condition (the resetting of the addresses from past LWA+1 to the FWA). In the WORKER the test for checking that the address is not past the end of the FIFO is done before the data is stored. In the FFT/FIT code this test is done after the previous location is accessed, as soon as the pointer is incremented. Thus if an FFT is called when OLDEST (the pointer to the start of the FFT data) points past the LWA+1 this will never be detected. The result is that once every 512 FFTs (on average) the system outputs garbage.

The 8085 Full Trace program was used to trap this condition (Section 6.5.5). The fix is to have the first task for either a P or an SFFT call check the OLDEST pointer for having been run past the end of the FIFO (to past LWA+1). The pointer is reset to the FIFO FWA if necessary. This fix is also implemented in the FFTMN program (Section 5.0.1).



## 7.0 PROGRAM VALIDATION

Once the program is debugged and error-free it is time to confirm correct operation. As was described in the Structured Programming Techniques Section (Section 3.6.1) the program was, as much as possible, partitioned into modular manageable sections which were then individually validated. The control flow between these sections was then verified. The Run Time code was essential in these activities.

In this fashion the entire program was confirmed to be operating correctly. In most cases this is a boring and time-consuming operation. However with the validation of the FFT some interesting outputs resulted. Contained in Appendix D is the Program Validation Output for the FFT operation. It is separated into two sections: (1) Subprogram Validation Series; and (2) Input/Output Validation Series.

### 7.1 Subprogram Validation Series

The Subprogram Validation Series demonstrates the operation of a PFFT with an Impulse Input and an SFFT with a DC Time Series Input. In both cases Output is listed after the operation of each of the various programs, concluding with the Magnitude Output. Operation on the Impulse Time Series Input results in a constant energy output for all frequency components. For a DC Time Series Input all the output energy is contained in the 0 Hz value. Correct operation of the Total Shift (TOTLSH) value was also validated but in the interest of clarity is not listed here.

## 7.2 Input/Output Validation Series

The Input/Output Validation Series demonstrates the output resulting from a set of boxcar function time inputs with widths ranging (1, 2, ...,  $N/4$ ,  $N/2$ , and  $N$ ). As expected the frequency outputs are  $(\sin(x)/x)^2$  functions with zeros located at (infinity,  $N/2$ , every  $N/4$ , ..., every 2, and all the energy at 0 Hz).

## 7.3 Frequency Input

Various frequency values were also input with the MODSIN function. As long as the indexes were even fractions of the sample frequency then impulse functions similar to the DC output resulted, located at the correct frequency index. If however frequencies were selected which were off these fractions then frequency leakage smeared the output spectra. Applying the SHAPE function improved the output, as expected. These outputs are not included in this report.

## 8.0 OPERATIONAL RESULTS

The FFT/FIT program is now debugged and verified, but how does it operate?

### 8.1 FFT Processing Times

Listed below are the processing times for the P and SFFTs. They have been measured fairly inaccurately (using a stopwatch), with Best times taken for the PFFT and a Range of Times taken for the SFFT. The times are quick enough to be useful, in that they put a very small limitation on the amount of time the ASP must be off line (and are also small compared to the processing time of the BOSS).

N	FFT Processing Times	
	PFFT	SFFT Range
64	0.64	
128	1.40	1.36 - 1.72
256	3.55	3.10 - 3.80
512		7.00 - 8.75

The times are plotted in Figure 8.1.a. As expected these times are proportional to the  $N \cdot \log_2(N)$  processing requirement characteristic of the FFT algorithm.

### 8.2 ASP System Results

(Quoting from [McEv]). The ASP System was first deployed in September through October of 1980 in The Geysers steam field in Northern California. As of August 1981 it had also been deployed in three other fields. Another ASP application has been the monitoring of thermally generated acoustic emissions associated with underground storage of nuclear wastes [Maje81]. This experiment demonstrated the flexibility of ASP in applying earthquake processing techniques

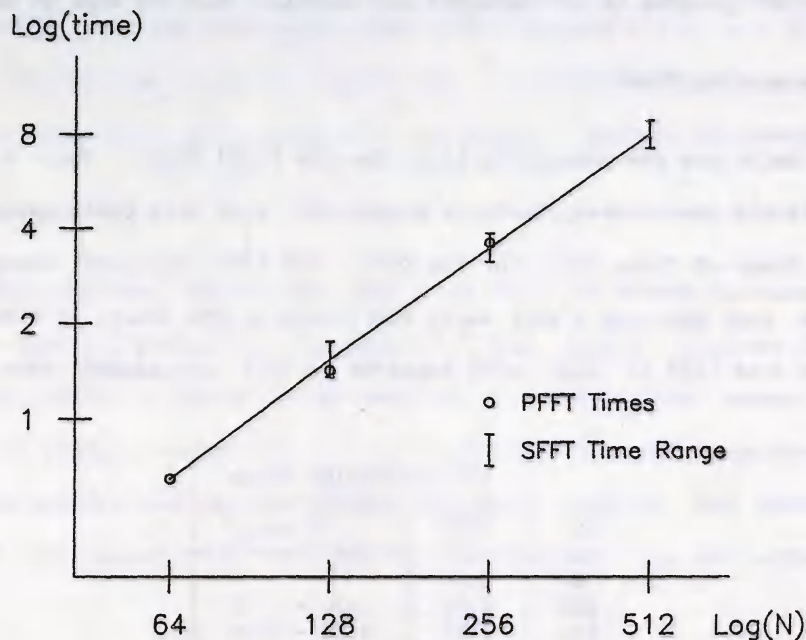


Figure 8.1.a FFT Processing Times

to a variety of different scales of interest.

Since that time a second, lab based system has been built for use by LBL geologists. In addition the technology developed for the ASP has been transitioned to industry. Both the hardware and the software are utilized in a commercial unit marketed by Sprengnether Instruments Inc. of St. Louis, MO. A number of these units have been built and sold to customers. The FFT/FIT code was used in both of these systems without any modification.

Throughout this time the FFT/FIT code has operated flawlessly in all of the units. No errors have resulted and no anomalies have been attributed to its operation. The code appears to fulfill the requirement for which it was developed.



## 9.0 SUMMARY

### 9.1 Results

(Quoting from [McEv]). Overall, the ASP concept has proved to be well-founded, and is applicable in a variety of situations. Furthermore, throughout the experiments described, which followed a thorough burn-in program, only one hardware failure has occurred in a PROM chip of one of the WORKERS. The field system, mounted in a truck, has been in a variety of weather and dust conditions and transported large distances with no adverse effects. The low-power DC operation (1 watt/channel), coupled with hardware reliability, allows remote field operation for extended periods of time.

The microearthquake processing software developed for ASP is relatively simple, requiring accurate data for effective usage. Our attention to accuracy in timing, amplitude, and spectral algorithms was absolutely necessary to the successful implementation of ASP. With processing time now reduced to less than 1 min/earthquake, the goals of cost-effectiveness and real-time results have been attained with no compromise in quality.

## 9.2 Conclusion

The FFT/FIT program has been integrated into an Automated Seismic Processor System for Microearthquake Networks and is being used to calculate the Power Spectral Density of earthquake energy for each of a number of seismometers. The resultant spectra is FIT for Long Period Level, Corner Frequency, and High Frequency Slope. These values are available to the geologists and geophysicists in an on-site, real-time mode and are of great assistance in their studies of microearthquakes and other seismic activities. The overall system has operated without significant difficulty for over 5 years and has proved a useful and valuable tool.

## 10.0 REFERENCES

[Alle] Allen, R.V. and J.O. Ellis, "An automated on-line monitor for microearthquake networks", EOS 61, 1030, 1980.

[Brig] Brigham, E. Oran, The Fast Fourier Transform, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1974.

[Kane] Kaneko, T. and B.Liu, "Accumulation of round-off errors in fast Fourier transforms", J. Assn. Comp. Mach., 17, no. 4, pp. 637-654, October 1970.

[Maje78] Majer, E.L., "Seismological investigations in geothermal regions", Lawrence Berkeley Lab., Rept. No. 7054, 225 pp., 1978.

[Maje79] Majer, E.L. and T.V. McEvelly, "Seismological investigations in The Geysers geothermal field", Geophysics 44, pp. 246-269, 1979.

[Maje80] Majer, E.L., T.V. McEvelly, A. Albores, and S. Diaz, "Seismological studies at the Cerro Prieto geothermal field", Geothermics 9, pp. 79-89, 1980.

[Maje81] Majer, E.L., T.V. McEvelly, and M.S. King, "Monitoring an underground repository with modern seismological methods", Int. Rock Mech. and Min. Sci., 18, pp. 517-527, 1981.

[McEv] McEvelly, T.V. and E.L. Mayer, "ASP: An Automated Seismic Processor for microearthquake networks", Bulletin of the Seismological Society of America, vol. 72, No. 1, pp. 303-325, February 1982.

[Oppe72] Oppenheim, Alan V. and C.J. Weinstein, "Effects of finite register length in digital filtering and the fast Fourier transform", Proc. IEEE, 60, No. 8, pp. 957-976, August 1972.

[Oppe75] Oppenheim, Alan V. and Ronald W. Schaffer, Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975, pp. 444-462.

[Rabi] Rabiner, L. R. and B. Gold, Theory and Application of Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975, pp. 587-594.

[Rich] Richter, C.F., Elementary Seismology, Freeman Press.

[Stew] Stewart, S.W., "Real-time detection and location of local events in central California", Bull. Seism. Soc. Am. pp. 433-452, 1977.

[Wein69a] Weinstien, Clifford J., "Quantization effects in digital filters", MIT Lincoln Lab., Tech. Report 468, ASTIA DOC. DDC AD-706862, Nov. 21, 1969.

[Wein69b] Weinstien, Clifford J., "Roundoff noise in floating point fast Fourier transform computation", IEEE Trans. Audio and Electroacoustics, vol. AU-17, pp. 209-215, September 1969.

[Welc] Welch, Peter D., "A fixed-point fast Fourier transform error analysis", IEEE Trans. Audio and Electroacoustics, vol. AU-17, pp. 162-164, June 1969.

ASP FFT Program  
Program Code

```

0000 ;          0026          ..BUSS
0000 ;          0027          XFWA=BBUFF          ..XMIT BUFFER 1ST WORD ADDR
0000 ;          0028          XLWA=XFWA+127
0000 ;          0029          RFWA=XLWA+1          ..RECEIVE BUFF FWA
0000 ;          0030          RLWA=RFWA+127
0000 ;          0031          ..UART
0000 ;          0032          IFWA=RLWA+1          ..INPUT BUFFER
0000 ;          0033          ILWA=IFWA+79
0000 ;          0034          OFWA=ILWA+1          ..OUTPUT BUFFER
0000 ;          0035          OLWA=OFWA+79
0000 ;          0036          ..MISC
0000 ;          0037          SFWA=OLWA+1          ..STRING BUFFER
0000 ;          0038          SLWA=SFWA+79
0000 ;          0039
0000 ;          0040
0000 ;          0041          ..REGISTER NAMES
0000 ;          0042
0000 ;          0043          ZAP=#00          ..RO IS SCRATCH PAD
0000 ;          0044          PC=#03          ..NORMAL PROG COUNTER
0000 ;          0045          DSP=#07          ..DATA STACK POINTER
0000 ;          0046          FRP=#08          ..FAST RAM POINTER
0000 ;          0047          MP=#09          ..MESSAGE POINTER
0000 ;          0048
0000 ;          0049          ZAP2=#0A
0000 ;          0050          ZAP3=#0B
0000 ;          0051          ZAP1=#0C          ..ARITH SCTATCHPAD
0000 ;          0052          MA=#0D          ..ARITH MEM ADDR
0000 ;          0053          MQ=#0E          ..ARITH ACC EXTENSION
0000 ;          0054          AC=#0F          ..ARITH ACCUMULATOR
0000 ;          0055
0000 ;          0056
0000 ;          0057
0000 ;          0058          ..*****
0000 ;          0059          ..WSYMBOLS          SYMBOLS USED BY WORKER
0000 ;          0060          ..*****
0000 ;          0061
0000 ;          0062
0000 ;          0063          SLAST=#3046          ..TIME FOR LAST YSFIFO
0000 ;          0064          DT=SLAST+2          ..ST MINUS PT
0000 ;          0065
0000 ;          0066
0000 ;          0067          ..FIFOS ARE DEFINED BY 9 BYTE TABLES AS FOLLOWS:
0000 ;          0068          ....BYTES 1,2          ADDRESS OF OLDEST DATA (H,L)
0000 ;          0069          ....BYTES 3,4          FIFO LWA+1 (H,L)
0000 ;          0070          ....BYTES 5,6          FIFO FWA (H,L)
0000 ;          0071          ....BYTES 7,8,9          FIFO RUNNING SUM (L,M,H)
0000 ;          0072
0000 ;          0073          YPTBL=#30A5          ..YPFIFO TABLE
0000 ;          0074          YSTBL=YPTBL+9          ..YSFIFO TABLE
0000 ;          0075
0000 ;          0076

```



```

0000 ;      0077  ..FIFOS THEMSELVES
0000 ;      0078
0000 ;      0079      YPFWA=#3601      ..YPFIFO
0000 ;      0080      YPLWA=YPFWA+1023
0000 ;      0081      YSFWA=YPLWA+1      ..YSFIFO
0000 ;      0082      YSLWA=YSFWA+1023
0000 ;      0083
0000 ;      0084
0000 ;      0085      ..(HEADER)
0000 ;      0086  ..*****
0000 ;      0087      ..(ALSO)
0000 ;      0088  ..EXTERNAL ROUTINUES
0000 ;      0089
0000 ;      0090      CALL=#0033      ..CALL
0000 ;      0091      RETN=#0043      ..RETURN
0000 ;      0092      INTP=#0050      ..INTERRUPT
0000 ;      0093      XBUSS=#010C      ..BUSS TRANSMIT
0000 ;      0094      MOVE=#0115      ..BLOCK MOVE
0000 ;      0095      BMESS=#0118      ..BUSS MESSAGE FORM
0000 ;      0096
0000 ;      0097
0000 ;      0098  ..*****
0000 ;      0099      ..(NEW)
0000 ;      0100
0000 ;      0101
0000 ;      0102      ..*****
0000 ;      0103      ..FFTSYMBOLS      SYMBOLS USED IN WORKER FFT
0000 ;      0104      ..*****
0000 ;      0105
0000 ;      0106
0000 ;      0107  ..*****
0000 ;      0108  ..***** IN FAST RAM (FRAM) *****
0000 ;      0109  ..*****
0000 ;      0110
0000 ;      0111  ..FLAGS
0000 ;      0112
0000 ;      0113      PFTFLG=YSTBL+22      ..P-FFT/S-FFT FLAG
0000 ;      0114      BDRYOK=PFTFLG+1      ..BOUNDARY OK FLAG
0000 ;      0115
0000 ;      0116  ..MOVE VARIABLES
0000 ;      0117
0000 ;      0118      FREELN=BDRYOK+1      ..FREE LENGTH
0000 ;      0119      FWA=FREELN+2      ..FWA OF MOVE OUT FIFO
0000 ;      0120      MOVECT=FWA+2      ..MOVE COUNT
0000 ;      0121
0000 ;      0122  ..LENGTH VALUES
0000 ;      0123
0000 ;      0124      PFFTLN=MOVECT+2      ..P-FFT LENGTH
0000 ;      0125      SFFTLN=PFFTLN+2      ..S-FFT LENGTH
0000 ;      0126      FFTLN=SFFTLN+2      ..FFT LENGTH
0000 ;      0127

```

ASP FFT Program  
Program Code

```

0000 ;          0128 ..LENGTH POINTERS
0000 ;          0129
0000 ;          0130          PLNPTR=FFTLN+2  ..P-LENGTH POINTER
0000 ;          0131          SLNPTR=PLNPTR+2  ..S-LENGTH POINTER
0000 ;          0132          FLNPTR=SLNPTR+2  ..FFT-LENGTH POINTER
0000 ;          0133
0000 ;          0134 ..START ADDRESS FOR FFT DATA
0000 ;          0135
0000 ;          0136          BASE=FLNPTR+2
0000 ;          0137
0000 ;          0138 ..SCALE VALUES
0000 ;          0139
0000 ;          0140          TOTLSH=BASE+2  ..TOTAL SHIFT VALUE
0000 ;          0141          PSHIFT=TOTLSH+2  ..P-FIFO SHIFT VALUE
0000 ;          0142
0000 ;          0143 ..SHAPE/FFT VALUE
0000 ;          0144
0000 ;          0145          CSVLU=PSHIFT+2  ..COSINE VALUE
0000 ;          0146
0000 ;          0147
0000 ;          0148 ..*****
0000 ;          0149 ..***** IN ROM *****
0000 ;          0150 ..*****
0000 ;          0151
0000 ;          0152
0000 ;          0153 ..***** LENGTH SET TABLES *****
0000 ;          0154
0000 ;          0155 ..THE LENGTH SET TABLES ARE DEFINED BY 10 BYTE
0000 ;          0156 ..DIRECTORIES AS FOLLOWS:
0000 ;          0157 ....BYTES 0,1  DT LENGTH BOUNDARY
0000 ;          0158 ....BYTES 2,3  P-LENGTH VALUE
0000 ;          0159 ....BYTES 4,5  S-LENGTH VALUE
0000 ;          0160 ....BYTES 6,7  POINTER INTO P-SHAPE TABLE
0000 ;          0161 ....BYTES 8,9  POINTER INTO S-SHAPE TABLE
0000 ;          0162
0000 ;          0163          FFTTBL=#2660  ..FFT TABLES (2660-27FF)
0000 ;          0164
0000 ;          0165          PDEFLT=FFTTBL  ..SOLO P-WAVE DEFAULT
0000 ;          0166
0000 ;          0167          DT128=PDEFLT+A  ..BOUNDARY TABLES
0000 ;          0168          DT600=DT128+A
0000 ;          0169          DTMAX=DT600+A
0000 ;          0170
0000 ;          0171
0000 ;          0172

```

```

0000 ;          0173 ..***** P & S SHAPE TABLES *****
0000 ;          0174
0000 ;          0175 ..FIRST THE P-SHAPE FUNCTION
0000 ;          0176 ....BYTES 0,1   THE COUNT FOR 75% MOVE
0000 ;          0177 ....BYTES 2,3   COUNT FOR 25% TAIL SHAPE
0000 ;          0178 ....BYTES 4,5   SKIP VALUE FOR TRIG TABLE
0000 ;          0179
0000 ;          0180 ..THERE WILL BE ONE TABLE FOR EACH LENGTH P-FFT
0000 ;          0181 ..          (64,128,256,512)
0000 ;          0182
0000 ;          0183          PLN64=DTMAX+A
0000 ;          0184          PLN128=PLN64+6
0000 ;          0185          PLN256=PLN128+6
0000 ;          0186          PLN512=PLN256+6
0000 ;          0187
0000 ;          0188
0000 ;          0189 ..THEN THE S-SHAPE FUNCTION
0000 ;          0190 ....BYTES 0,1   COUNT FOR 12.5% FRONT SHAPE
0000 ;          0191 ....BYTES 2,3   SKIP VALUE FOR 12.5% FRONT
0000 ;          0192 ....BYTES 4,5   COUNT FOR 62.5% MIDDLE
0000 ;          0193 ....BYTES 6,7   COUNT FOR 25% TAIL SHAPE
0000 ;          0194 ....BYTES 8,9   SKIP VALUE FOR 25% REAR
0000 ;          0195
0000 ;          0196 ..THERE WILL BE ONE TABLE FOR EACH LENGTH S-FFT
0000 ;          0197 ..          (128,256,512)
0000 ;          0198
0000 ;          0199          SLN128=PLN512+6
0000 ;          0200          SLN256=SLN128+10
0000 ;          0201          SLN512=SLN256+10
0000 ;          0202
0000 ;          0203
0000 ;          0204
0000 ;          0205 ..***** TRIG TABLE DIRECTORY *****
0000 ;          0206
0000 ;          0207 ..THE TRIG DATA TABLE IS DEFINED
0000 ;          0208 ..WITH A 4 BYTE DIRECTORY AS SHOWN BELOW:
0000 ;          0209
0000 ;          0210          TRGDIR=SLN512+10
0000 ;          0211
0000 ;          0212 ..TRIG TABLE DIRECTORY CONTENTS
0000 ;          0213
0000 ;          0214          TRGLWA=#26FE+256 ..TRIG LAST WORD ADDRESS
0000 ;          0215          TRGFWA=#26FE    ..TRIG FIRST WORD ADDRESS
0000 ;          0216
0000 ;          0217

```

ASP FFT Program  
Program Code

```

0000 ; 0218 ..***** BIT MASK TABLE *****
0000 ; 0219
0000 ; 0220 ..THE BIT MASK TABLE IS USED IN THE FFT ALGORITHM
0000 ; 0221 ..IT CONSISTS OF 10 ENTRIES
0000 ; 0222 ..0400, 0200, ..., 0040, 0000
0000 ; 0223
0000 ; 0224             BITTBL=TRGDIR+4
0000 ; 0225
0000 ; 0226
0000 ; 0227
0000 ; 0228 ..***** FFT LENGTH TABLES *****
0000 ; 0229
0000 ; 0230 ..THESE TABLES ARE USED DURING THE FFT ALGORITHM
0000 ; 0231 ...BYTES 0,1   POINTER TO THE BIT MASK TABLE
0000 ; 0232 ...BYTES 2,3   SHIFT COUNT FOR BITREV ROUTINE
0000 ; 0233 ...AND MORE
0000 ; 0234
0000 ; 0235             FLN512=BITTBL+20
0000 ; 0236             FLN256=FLN512+8
0000 ; 0237             FLN128=FLN256+8
0000 ; 0238             FLN64=FLN128+8
0000 ; 0239
0000 ; 0240

```



## A.2 FFT/FIT CODE

### A.2.0.1 FFTMN (FFT Main)

```

0000 ;          0001
0000 ;          0002 ..FFTMN.SR          24 MAY 80          4:15 PM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..          FFTMN.SR
0000 ;          0244 ..
0000 ;          0245 ..THIS IS THE CALLING PROGRAM FOR CONTROL OF THE
0000 ;          0246 ..SPECTRA COMPUTATION AND FITTING PROGRAM
0000 ;          0247 ..
0000 ;          0248 ..IT HAS TWO ENTRY POINTS:
0000 ;          0249 ..      1. P-FFT CALL, OR
0000 ;          0250 ..      2. S-FFT CALL
0000 ;          0251 ..AND IT THEN BRANCHES FROM THERE TO THE VARIOUS
0000 ;          0252 ..SUBROUTINES
0000 ;          0253 ..#####
0000 ;          0254
0000 ;          0255
0000 ;          0256
0000 ;          0257 ..EXTERNAL ROUTINES
0000 ;          0258          LNSET=#17A0          ..LENGTH SET
0000 ;          0259          PLNSET=LNSET          ..P-LENGTH SET
0000 ;          0260          SLNSET=LNSET+3          ..S-LENGTH SET
0000 ;          0261          SCALE=#18A0          ..SCALE
0000 ;          0262          PSCALE=SCALE          ..P-FFT SCALE
0000 ;          0263          SSCALE=SCALE+3          ..S-FFT SCALE
0000 ;          0264          SHAPE=#1900          ..SHAPE
0000 ;          0265          PSHAPE=SHAPE          ..P-FFT SHAPE
0000 ;          0266          SSHAPE=SHAPE+3          ..S-FFT SHAPE
0000 ;          0267          FFT=#1A60          ..FFT ALGORITHM
0000 ;          0268          UNSCRM=#1C00          ..UNSRAMBLE
0000 ;          0269          ORDER=#1D20          ..COLUMN REORDER
0000 ;          0270          MAGAPX=#2000          ..COMPUTE MAGNITUDE SQUARED
0000 ;          0271          SMTGMX=#2100          ..SMOOTH AND MAXIMUM TAG
0000 ;          0272          CKFMAX=#2250          ..CHECK FMAX FOR VALIDITY
0000 ;          0273          GMACPT=#23F0          ..GAMMA COMPUTE
0000 ;          0274          FOLPL=#2C00          ..FO & LPL COMPUTE
0000 ;          0275

```



ASP FFT Program  
Program Code

```

0000 ;          0276 ..INTERNAL VARIABLES
0000 ;          0277
0000 ;          0278          LNFFT=#30E8          ..LENGTH OF FFT
0000 ;          0279          FO=LNFFT+2          ..CORNER FREQUENCY
0000 ;          0280          GMAINT=FO+2          ..GAMMA INTEGER
0000 ;          0281          GMAFRC=GMAINT+2      ..GAMMA FRACTION
0000 ;          0282          MXNINT=GMAFRC+2      ..MAXIMUM ENERGY INTEGER
0000 ;          0283          MXNEXP=MXNINT+2      ..MAXIMUM ENERGY EXPONENT
0000 ;          0284          ..(CORRECTED MAX VALUE,
0000 ;          0285          ..THEN LONG PERIOD LEVEL)
0000 ;          0286
0000 ;          0287          ORG #1700
1700 ;          0288
1700 ;          0289 ..*****
1700 ;          0290 ..BEGIN HERE
1700 ;          0291
1700 C01706;    0292          LBR PFFT          ..GOTO P-FFT
1703 C01716;    0293          LBR SFFT          ..GOTO S-FFT
1706 ;          0294
1706 ;          0295
1706 ;          0296 ..*****
1706 ;          0297 PFFT:          ..P-FFT CALL
1706 ;          0298 ..GET HIGH R(FRP) & R(ZAP1) TO FRAM
1706 F830B8BC;  0299          LDI A.1(FRAM); PHI FRP; PHI ZAP1
170A ;          0300
170A ;          0301 ..CALL P-LENGTH SET
170A D417A0;    0302          CALL PLNSET
170D ;          0303
170D ;          0304 ..CALL P-FFT SCALE
170D D418A0;    0305          CALL PSCALE
1710 ;          0306
1710 ;          0307 ..JUMP OUT TO 1ST PATCH
1710 C01780;    0308          LBR PATCH1
1713 ;          0309
1713 ;          0310 ..BUT REMEMBER WHERE YOU CAME FROM
1713 ;          0311 MAIN1:          ..PATCH #1 RETURN ADDRESS
1713 ;          0312
1713 ;          0313 ..AT 1ST PATCH SET SHOLD
1713 ;          0314 ..SO ADC DOESN'T OVERWRITE YSFIFO
1713 ;          0315          ORG #1780
1780 F825A8;    0316 PATCH1: LDI #25; PLO FRP          ..#25 = A.0(SHOLD)
1783 F8FF58;    0317          LDI #FF; STR FRP
1786 ;          0318
1786 ;          0319 ..CALL P-SHAPE FROM HERE
1786 D41900;    0320          CALL PSHAPE
1789 ;          0321
1789 ;          0322 ..THEN RETURN TO MAIN BODY
1789 C01713;    0323          LBR MAIN1
178C ;          0324

```

ASP FFT Program  
Program Code

```

178C ;      0325 ..CONTINUE
178C ;      0326         ORG MAIN1
1713 ;      0327
1713 ;      0328 ..JUMP TO FFT COMMON
1713 C01723; 0329         LBR FFTCOM
1716 ;      0330
1716 ;      0331
1716 ;      0332 ..*****
1716 ;      0333 SFFT:          ..S-FFT CALL
1716 ;      0334 ..GET HIGH R(FRP) & R(ZAP1) TO FRAM
1716 F830B8BC; 0335         LDI A.1(FRAM); PHI FRP; PHI ZAP1
171A ;      0336
171A ;      0337 ..CALL S-LENGTH SET
171A D417A3;  0338         CALL SLNSET
171D ;      0339
171D ;      0340 ..CALL S-FFT SCALE
171D D418A3;  0341         CALL SSCALE
1720 ;      0342
1720 ;      0343 ..CALL S-FFT SHAPE
1720 D41903;  0344         CALL SSHAPE
1723 ;      0345
1723 ;      0346 ..AND FALL THROUGH TO FFT COMMON
1723 ;      0347
1723 ;      0348 ..*****
1723 ;      0349 FFTCOM:          ..FFT COMMON
1723 ;      0350 ..CALL FFT ALGORITHM
1723 D41A60;   0351         CALL FFT
1726 ;      0352
1726 ;      0353 ..CALL UNSCRAMBLE ROUTINUE
1726 D41C00;  0354         CALL UNSCRM
1729 ;      0355
1729 ;      0356 ..CALL COLUMN REORDER
1729 D41D20;  0357         CALL ORDER
172C ;      0358
172C ;      0359 ..CALL MAGNITUDE APPROXIMATE
172C D42000;  0360         CALL MAGAPX
172F ;      0361
172F ;      0362 ..CALL SMOOTH & MAXIMUM ROUTINUE
172F D42100;  0363         CALL SMTGMX
1732 ;      0364
1732 ;      0365 ..CALL ROUTINUE TO CHECK FOR FMAX VALID,
1732 D42250;  0366         CALL CKFMAX
1735 ;      0367
1735 ;      0368 ..ON RETURN, BRANCH OUT TO PATCH #2
1735 3073;    0369         BR PATCH2
1737 ;      0370
1737 ;      0371 ..BUT REMEMBER WHERE YOU CAME FROM
1737 ;      0372 MAIN2:          ..PATCH #2 RETURN ADDRESS
1737 ;      0373

```

ASP FFT Program  
Program Code

1737 ;	0374 ..FROM CKFMAX, IF FMAX INVALID THEN R(AC.0) NOT #00
1737 ;	0375 ORG #1773
1773 8F;	0376 PATCH2: GLO AC
1774 CA173D;	0377 LBNZ TRRSLT ..IF NO GOOD, THEN
1777 ;	0378 ..GOTO TRANSMIT RESULTS
1777 ;	0379 ..(WITH ERROR FLAG)
1777 ;	0380
1777 ;	0381 ..ELSE RETURN TO BODY OF PROGRAM
1777 C01737;	0382 LBR MAIN2
177A ;	0383
177A ;	0384 ..CONTINUE
177A ;	0385 ORG MAIN2
1737 ;	0386
1737 ;	0387 ..CALL GAMMA COMPUTE
1737 D423F0;	0388 CALL GMACPT
173A ;	0389
173A ;	0390 ..CALL FO & LPL COMPUTE
173A D42C00;	0391 CALL FOLPL
173D ;	0392
173D ;	0393 TRRSLT: ..TRANSMIT RESULTS
173D ;	0394 ..CALL TO TRANSMIT RESULTS
173D ;	0395 ..CHECK FOR P-FFT
173D F8C4A8;	0396 LDI A.0(PFTFLG); PLO FRP
1740 08;	0397 LDN FRP
1741 3259;	0398 BZ SNDSFT ..IF IT'S A S-FFT, THEN
1743 ;	0399 ..GOTO SEND S-FFT RESULTS
1743 ;	0400
1743 ;	0401 ..ELSE IT'S A P-FFT, GET MESSAGE READY
1743 D40118;	0402 CALL BMESS
1746 06;	0403 ,#06
1747 46;	0404 ,T'F'
1748 30E8;	0405 ,A(LNFFT)
174A 30EA;	0406 ,A(FO)
174C 30F0;	0407 ,A(MXNINT)
174E 30F2;	0408 ,A(MXNEXP)
1750 30EC;	0409 ,A(GMAINT)
1752 30EE;	0410 ,A(GMAFRC)
1754 ;	0411
1754 ;	0412 ..SEND RESULTS, THEN EXIT TO WORKER
1754 D4010C;	0413 CALL XBUSS
1757 ;	0414
1757 ;	0415 ..THEN BRANCH OUT TO 3RD PATCH
1757 3090;	0416 BR PATCH3
1759 ;	0417
1759 ;	0418 ..BUT REMEMBER WHERE YOU CAME FROM
1759 ;	0419 MAIN3: ..PATCH #3 RETURN ADDRESS
1759 ;	0420

```

1759 ;          0421 ..SINCE END OF P-FFT COMPUTE, RELEASE SHOLD
1759 ;          0422      ORG #1790
1790 F825A8;      0423 PATCH3: LDI #25; PLO FRP          ..#25 = A.0(SHOLD)
1793 F80058;      0424      LDI #00; STR FRP
1796 ;          0425
1796 ;          0426 ..THEN RETURN TO MAIN (TO RETURN TO WORKER)
1796 306D;        0427      BR XFFTMN
1798 ;          0428
1798 ;          0429 ..CONTINUE
1798 ;          0430      ORG MAIN3
1759 ;          0431
1759 ;          0432 SNDSFT:          ..SEND S-FFT RESULTS
1759 ;          0433 ..GET MESSAGE READY
1759 D40118;      0434      CALL BMESS
175C 06;         0435      ,#06
175D 47;         0436      ,T'G'
175E 30E8;       0437      ,A(LNFFT)
1760 30EA;       0438      ,A(FO)
1762 30F0;       0439      ,A(MXNINT)
1764 30F2;       0440      ,A(MXNEXP)
1766 30EC;       0441      ,A(GMAINT)
1768 30EE;       0442      ,A(GMAFRC)
176A ;          0443
176A ;          0444 ..SEND S-FFT MESSAGE, THEN RETURN TO WORKER
176A D4010C;     0445      CALL XBUSS
176D ;          0446
176D ;          0447 ..-----
176D ;          0448 XFFTMN:          ..EXIT FFT MAIN
176D ;          0449 ..RETURN TO WORKER
176D D5;         0450      EXIT
176E C4C4C4;     0451      ,#C4C4C4
1771 ;          0452
1771 ;          0453 ..#####
1771 ;          0454 ..END OF FFTMN.SR

```



ASP FFT Program  
Program Code

A.2.1 FFT Code

A.2.1.1 LNSET (Length Set)

```

0000 ;          0001
0000 ;          0002 ..LNSET.SR          24 MAY 80          7:00 PM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..          LNSET.SR
0000 ;          0244 ..
0000 ;          0245 ..THIS PROGRAM SETS THE P-FFT FLAG, THE LENGTH
0000 ;          0246 ..          VALUES, THE LENGTH POINTERS, AND THE
0000 ;          0247 ..          BASE VALUE FOR THE FFT ALGORITHM
0000 ;          0248 ..
0000 ;          0249 ..IF A P-FFT HAS BEEN CALLED AND THERE HAS BEEN
0000 ;          0250 ..NO S-WAVE DETECTION THEN THE DEFAULT CONDITION
0000 ;          0251 ..IS SET:
0000 ;          0252 ..          P-LENGTH = 128
0000 ;          0253 ..
0000 ;          0254 ..OTHERWISE THE RULES ARE:
0000 ;          0255 ..          DT <= 128          P-LENGTH=64          S-LENGTH=128
0000 ;          0256 ..128 < DT <= 600          P-LENGTH=128          S-LENGTH=256
0000 ;          0257 ..600 < DT <= 32K          P-LENGTH=256          S-LENGTH=512
0000 ;          0258 ..#####
0000 ;          0259
0000 ;          0260
0000 ;          0261 ..VECTORS TO INTERNAL ROUTINES
0000 ;          0262          ORG #17A0
17A0 C0187C; 0263          LBR PATCH1          ..W/PLNSET, CHECK YP-OLDEST
17A3 C0188E; 0264          LBR PATCH2          ..W/SLNSET, CHECK YS-OLDEST
17A6 ;          0265
17A6 ;          0266 ..BUT REMEMBER WHERE YOU CAME FROM
17A6 ;          0267 MAIN12:          ..TO CONTINUE PROGRAM
17A6 ;          0268
17A6 ;          0269 ..WITH PLNSET, CHECK YP-OLDEST FOR YPLWA+1
17A6 ;          0270          ORG #187C
187C F8A5A8; 0271 PATCH1: LDI A.0(YPTBL); PLO FRP
187F 08FF3A; 0272          LDN FRP; SMI A.1(YPLWA+1)
1882 CB17A6; 0273          LBNF PLNSET          ..IF < YPLWA+1, THEN
1885 ;          0274          ..GOTO SET LENGTH
1885 ;          0275
1885 ;          0276 ..ELSE FORCE TO YPFWA (CORRECT VALUE)
1885 F83658; 0277          LDI A.1(YPFWA); STR FRP
1888 C017A6; 0278          LBR PLNSET          ..THEN GOTO SET LENGTH
188B ;          0279

```



```

188B ;          0280 ..WITH SLNSET, CHECK YS-OLDEST FOR YSLWA+1
188B ;          0281      ORG #188E
188E F8AEA8;    0282 PATCH2: LDI A.0(YSTBL); PLO FRP
1891 08FF3E;    0283      LDN FRP; SMI A.1(YSLWA+1)
1894 CB17C1;    0284      LBNF SLNSET      ..IF < YSLWA+1, THEN
1897 ;          0285                  ..GOTO SET LENGTH
1897 ;          0286
1897 ;          0287 ..ELSE FORCE TO YSFWA (CORRECT VALUE)
1897 F83A58;    0288      LDI A.1(YSFWA); STR FRP
189A C017C1;    0289      LBR SLNSET      ..THEN GOTO SET LENGTH
189D ;          0290
189D ;          0291 ..CONTINUE WITH MAIN PROGRAM
189D ;          0292      ORG MAIN12
17A6 ;          0293
17A6 ;          0294 ..#####
17A6 ;          0295      ..          PLNSET.SR
17A6 ;          0296      ..
17A6 ;          0297 ..YOU ENTER HERE IF AN P-FFT HAS BEEN CALLED
17A6 ;          0298      ..
17A6 ;          0299 ..THIS SECTION CHECKS IF AN S-WAVE WAS DETECTED
17A6 ;          0300 ..BY LOOKING AT SLAST
17A6 ;          0301      ..
17A6 ;          0302 ..IF      SLAST = 0 , NO S-WAVE
17A6 ;          0303      ..          AND P-LENGTH=256 (DEFAULT)
17A6 ;          0304 ..ELSE (SLAST <> 0), YOU CONTINUE TO THE P & S
17A6 ;          0305      ..          LENGTH SET ROUTINUE (SLNSET)
17A6 ;          0306 ..#####
17A6 ;          0307
17A6 ;          0308 PLNSET:          ..P-LENGTH SET (SOLO)
17A6 ;          0309
17A6 ;          0310 ..SET P-FFT FLAG TRUE
17A6 F8C4;      0311      LDI A.0(PFTFLG)
17A8 AC;        0312      PLO ZAP1
17A9 F8FF;      0313      LDI #FF
17AB 5C;        0314      STR ZAP1
17AC ;          0315
17AC ;          0316 ..LOAD SLAST ADDRESS INTO R(FRP)
17AC F846;      0317      LDI A.0(SLAST)
17AE A8;        0318      PLO FRP
17AF ;          0319
17AF ;          0320 ..CHECK BYTES FOR NOT ZERO
17AF E8;        0321      SEX FRP          ..CHECK HIGH BYTE
17B0 72;        0322      LDXA
17B1 CA17C7;    0323      LBNZ DTCHK      ..JUMP OUT IF HAD S
17B4 72;        0324      LDXA          ..LOW TOO
17B5 CA17C7;    0325      LBNZ DTCHK      ..ETC
17B8 ;          0326

```

ASP FFT Program  
Program Code

```

17B8 ;          0327 ..SLAST IS ZERO, NO S, CHOOSE DEFAULT
17B8 ;          0328 ..LOAD P-DEFAULT LENGTH ADDR INTO R(MP)
17B8 F826;      0329             LDI A.1(PDEFLT+2)
17BA B9;        0330             PHI MP
17BB F862;      0331             LDI A.0(PDEFLT+2)
17BD A9;        0332             PLO MP
17BE ;          0333
17BE ;          0334 ..AND GO DOWN TO STORE POINTERS
17BE C017E8;    0335             LBR STPSLN
17C1 ;          0336
17C1 ;          0337
17C1 ;          0338
17C1 ;          0339 ..#####
17C1 ;          0340 ..             SLNSET.SR
17C1 ;          0341 ..
17C1 ;          0342 ..YOU ENTER HERE IF AN S-FFT HAS BEEN CALLED
17C1 ;          0343 ..OR (FURTHER DOWN) FOR CONTINUING THE P-FFT
17C1 ;          0344 ..             (NOT SELECTING DEFAULT)
17C1 ;          0345 ..
17C1 ;          0346 ..IT SETS P & S-LENGTH POINTERS
17C1 ;          0347 ..ACCORDING TO THE DT RULES ABOVE
17C1 ;          0348 ..#####
17C1 ;          0349
17C1 ;          0350 SLNSET:             ..S-LENGTH SET
17C1 ;          0351
17C1 ;          0352 ..SET P-FFT FLAG FALSE
17C1 F8C4;      0353             LDI A.0(PFTFLG)
17C3 AC;        0354             PLO ZAP1
17C4 F800;      0355             LDI #00
17C6 5C;        0356             STR ZAP1
17C7 ;          0357
17C7 ;          0358 ..LOAD DT ADDRESS INTO R(FRP)
17C7 ;          0359 DTCHK:             ..DT CHECK (P-SET CONTINUE)
17C7 F848;      0360             LDI A.0(DT)
17C9 A8;        0361             PLO FRP
17CA ;          0362
17CA ;          0363 ..LOAD DT INTO R(ZAP3)
17CA E8;        0364             SEX FRP
17CB 72;        0365             LDXA
17CC BB;        0366             PHI ZAP3
17CD 72;        0367             LDXA
17CE AB;        0368             PLO ZAP3
17CF ;          0369
17CF ;          0370 ..GET LOW BOUNDARY ADDRESS INTO R(MP)
17CF F826;      0371             LDI A.1(DT128)
17D1 B9;        0372             PHI MP
17D2 F86A;      0373             LDI A.0(DT128)
17D4 A9;        0374             PLO MP
17D5 ;          0375

```

```

17D5 ;          0376 ..CHECK FOR DT <= BOUNDARY
17D5 ;          0377 DTBR SB:          ..BDRY - DT SUBTRACT
17D5 E9;        0378          SEX MP
17D6 19;        0379          INC MP
17D7 8B;        0380          GLO ZAP3
17D8 F5;        0381          SD
17D9 29;        0382          DEC MP
17DA 9B;        0383          GHI ZAP3
17DB 75;        0384          SDB
17DC ;          0385
17DC ;          0386 ..GET AHEAD TWO PLACES
17DC 19;        0387          INC MP
17DD 19;        0388          INC MP
17DE ;          0389
17DE ;          0390 ..IF RESULT POSITIVE, STORE PLN & SLN VALUES
17DE C317E8;    0391          LBDF STPSLN
17E1 ;          0392
17E1 ;          0393 ..ELSE GET TO NEW BOUNDARY
17E1 89FC08A9;  0394          GLO MP; ADI #08; PLO MP
17E5 C017D5;    0395          LBR DTBR SB
17E8 ;          0396
17E8 ;          0397
17E8 ;          0398 ..*****
17E8 ;          0399 ..      STORE VALUES
17E8 ;          0400 ..*****
17E8 ;          0401
17E8 ;          0402 STPSLN:          ..STORE PLN & SLN VALUES
17E8 ;          0403
17E8 ;          0404 ..GET R(FRP) TO P-FFT LENGTH ADDRESS
17E8 F8CC;      0405          LDI A.0(PFFTLN)
17EA A8;        0406          PLO FRP
17EB ;          0407
17EB ;          0408 ..PASS P-FFT LENGTH VALUE
17EB 495818;    0409          LDA MP; STR FRP; INC FRP
17EE 495818;    0410          LDA MP; STR FRP; INC FRP
17F1 ;          0411
17F1 ;          0412 ..PASS S-FFT LENGTH
17F1 495818;    0413          LDA MP; STR FRP; INC FRP
17F4 495818;    0414          LDA MP; STR FRP; INC FRP
17F7 ;          0415
17F7 ;          0416 ..GET TO P-FFT LENGTH POINTER
17F7 1818;      0417          INC FRP; INC FRP
17F9 ;          0418
17F9 ;          0419 ..LOAD AND STORE PLNPTR
17F9 49;        0420          LDA MP          ..LOW
17FA 58;        0421          STR FRP
17FB 18;        0422          INC FRP
17FC 49;        0423          LDA MP          ..AND HIGH TOO
17FD 58;        0424          STR FRP
17FE 18;        0425          INC FRP
17FF ;          0426

```

ASP FFT Program  
Program Code

17FF ;	0427 ..WITH SLNPTR TOO	
17FF 49;	0428 LDA MP	..LOW
1800 58;	0429 STR FRP	
1801 18;	0430 INC FRP	
1802 49;	0431 LDA MP	..AND HIGH TOO
1803 58;	0432 STR FRP	
1804 18;	0433 INC FRP	
1805 ;	0434	
1805 ;	0435 ..GET R(FRP) TO BASE	
1805 1818;	0436 INC FRP; INC FRP	
1807 ;	0437	
1807 ;	0438 ..CHECK FOR P-FFT OR S-FFT	
1807 ;	0439 ..IF S-FFT BRANCH OUT	
1807 0C;	0440 LDN ZAP1	
1808 C21826;	0441 LBZ FISSFT	..FFT IS S-FFT
180B ;	0442	
180B ;	0443 ..ELSE P-FFT, SET BASE = YSFWA	
180B F83A5818;	0444 LDI A.1(YSFWA); STR FRP; INC FRP	
180F F80158;	0445 LDI A.0(YSFWA); STR FRP	
1812 ;	0446	
1812 ;	0447 ..MOVE P-FFT LENGTH VALUE INTO FFT LENGTH	
1812 F830B9;	0448 LDI A.1(PFFTLN); PHI MP	
1815 F8CCA9;	0449 LDI A.0(PFFTLN); PLO MP	
1818 49BF49AF;	0450 LDA MP; PHI AC; LDA MP; PLO AC	
181C 1919;	0451 INC MP; INC MP	
181E 9F5919;	0452 GHI AC; STR MP; INC MP	
1821 8F59;	0453 GLO AC; STR MP	
1823 ;	0454	
1823 ;	0455 ..JUMP DOWN TO SET FFT LENGTH POINTER	
1823 C0183D;	0456 LBR STFLPT	..SET FFT LENGTH POINTER
1826 ;	0457	
1826 ;	0458 FISSFT:	..FFT IS S-FFT
1826 ;	0459 ..SET BASE = S-OLDEST DATA ADDRESS	
1826 ;	0460 ..GET R(ZAP1) TO S-OLDEST POINTER	
1826 F8AEAC;	0461 LDI A.0(YSTBL); PLO ZAP1	
1829 ;	0462	
1829 ;	0463 ..PASS ADDRESS TO BASE	
1829 4C5818;	0464 LDA ZAP1; STR FRP; INC FRP	
182C 4C58;	0465 LDA ZAP1; STR FRP	
182E ;	0466	
182E ;	0467 ..MOVE S-FFT LENGTH VALUE INTO FFT LENGTH	
182E F830B9;	0468 LDI A.1(SFFTLN); PHI MP	
1831 F8CEA9;	0469 LDI A.0(SFFTLN); PLO MP	
1834 49BF49AF;	0470 LDA MP; PHI AC; LDA MP; PLO AC	
1838 9F5919;	0471 GHI AC; STR MP; INC MP	
183B 8F59;	0472 GLO AC; STR MP	
183D ;	0473	



```

183D ;      0474 STFLPT:                ..SET FFT LENGTH POINTER
183D ;      0475 ..FFT LENGTH IS IN R(AC)
183D ;      0476 ..IF FFT LENGTH = 512, FLNPTR = FLN512,
183D ;      0477 ..IF FFT LENGTH = 256, FLNPTR = FLN256, ETC.
183D ;      0478
183D ;      0479 ..GET R(MP) TO FLN512
183D F826B9; 0480         LDI A.1(FLN512); PHI MP
1840 F8D6A9; 0481         LDI A.0(FLN512); PLO MP
1843 ;      0482
1843 ;      0483 ..IF LENGTH = 512, THEN GOTO STORE
1843 8FFD00CA184F; 0484         GLO AC; SDI #00; LBNZ MB256
1849 9F7D04C21870; 0485         GHI AC; SDBI #04; LBZ SRFLPT
184F ;      0486
184F ;      0487 MB256:                ..MAYBE LENGTH 256
184F ;      0488 ..GET R(MP) TO FLN256
184F F8DEA9; 0489         LDI A.0(FLN256); PLO MP
1852 ;      0490
1852 ;      0491 ..IF LENGTH = 256, THEN GOTO STORE
1852 8FFD00CA185E; 0492         GLO AC; SDI #00; LBNZ MB128
1858 9F7D02C21870; 0493         GHI AC; SDBI #02; LBZ SRFLPT
185E ;      0494
185E ;      0495 MB128:                ..MAYBE LENGTH 128
185E ;      0496 ..GET R(MP) TO FLN128
185E F8E6A9; 0497         LDI A.0(FLN128); PLO MP
1861 ;      0498
1861 ;      0499 ..IF LENGTH = 128, THEN GOTO STORE
1861 8FFD00CA186D; 0500         GLO AC; SDI #00; LBNZ MB64
1867 9F7D01C21870; 0501         GHI AC; SDBI #01; LBZ SRFLPT
186D ;      0502
186D ;      0503 MB64:                ..MUST BE LENGTH 64
186D ;      0504 ..GET R(MP) TO FLN64
186D F8EEA9; 0505         LDI A.0(FLN64); PLO MP
1870 ;      0506 ..FALL THROUGH TO STORE
1870 ;      0507
1870 ;      0508 SRFLPT:                ..STORE FFT LENGTH POINTER
1870 ;      0509 ..POINTER AT R(MP) GOES TO FFT LENGTH POINTER
1870 F8D6AC; 0510         LDI A.0(FLNPTR); PLO ZAP1
1873 995C1C; 0511         GHI MP; STR ZAP1; INC ZAP1
1876 895C; 0512         GLO MP; STR ZAP1
1878 ;      0513
1878 ;      0514 ..AND RETURN
1878 D5; 0515         EXIT
1879 ;      0516
1879 ;      0517         END

```



ASP FFT Program  
Program Code

A.2.1.2 SCALE (Scale)

```

0000 ;          0001
0000 ;          0002 ..SCALE.SR          20 JAN 80          5:00 PM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..          SCALE.SR
0000 ;          0244 ..
0000 ;          0245 ..THIS ROUTINUE SCALES THE P-FFT OR THE S-FFT DATA
0000 ;          0246 ..TO THE MAXIMUM VIA USE OF THE DATA SCALE
0000 ;          0247 ..SUBROUTINUE
0000 ;          0248 ..IN THIS CASE THE MAXIMUM VALUES CAN EXTEND TO
0000 ;          0249 ..#7FFF OR TO #8000 SO MAXSHIFT IS 7
0000 ;          0250 ..#####
0000 ;          0251
0000 ;          0252
0000 ;          0253
0000 ;          0254 ..EXTERNAL ROUTINUES
0000 ;          0255          DTASCL=#2960          ..DATA SCALE
0000 ;          0256
0000 ;          0257
0000 ;          0258          ORG #18A0
18A0 ;          0259 ..BEGIN HERE
18A0 ;          0260
18A0 C018A6;      0261          LBR PSCALE          ..GOTO P-FFT SCALE
18A3 C018B2;      0262          LBR SSCALE          ..GOTO S-FFT SCALE
18A6 ;          0263
18A6 ;          0264
18A6 ;          0265 ..*****
18A6 ;          0266 PSCALE:          ..P-FFT SCALE
18A6 ;          0267 ..GET R(FRP) TO LENGTH VALUE
18A6 F8CCA8;      0268          LDI A.0(PFFTLN); PLO FRP
18A9 ;          0269
18A9 ;          0270 ..GET R(MP) TO P-OLDEST ADDRESS
18A9 F830B9;      0271          LDI A.1(YPTBL); PHI MP
18AC F8A5A9;      0272          LDI A.0(YPTBL); PLO MP
18AF ;          0273
18AF ;          0274 ..BRANCH TO SCALE COMMON
18AF C018BB;      0275          LBR SCLCOM
18B2 ;          0276
18B2 ;          0277

```

```

18B2 ;          0278  ..*****
18B2 ;          0279  SSCALE:                ..S-FFT SCALE
18B2 ;          0280  ..GET R(FRP) TO LENGTH VALUE
18B2 F8CEA8;    0281      LDI A.0(SFFTLN); PLO FRP
18B5 ;          0282
18B5 ;          0283  ..GET R(MP) TO S-OLDEST ADDRESS
18B5 F830B9;    0284      LDI A.1(YSTBL); PHI MP
18B8 F8AEA9;    0285      LDI A.0(YSTBL); PLO MP
18BB ;          0286
18BB ;          0287  ..FALL THROUGH TO SCALE COMMON
18BB ;          0288
18BB ;          0289  ..*****
18BB ;          0290  SCLCOM:                ..SCALE COMMON
18BB ;          0291  ..CLEAR TOTAL SHIFT VALUE
18BB F830BE;    0292      LDI A.1(TOTLSH); PHI MQ
18BE F8DAAE;    0293      LDI A.0(TOTLSH); PLO MQ
18C1 F8005E1E5E; 0294      LDI #00; STR MQ; INC MQ; STR MQ
18C6 ;          0295
18C6 ;          0296  ..GET R(ZAP1) TO LWA+1 BOUNDARY
18C6 89FC02AC;  0297      GLO MP; ADI #02; PLO ZAP1
18CA ;          0298
18CA ;          0299  ..LOAD MAXSHIFT, 7, INTO R(ZAP2)
18CA F800BAF807AA; 0300      LDI #00; PHI ZAP2; LDI #07; PLO ZAP2
18D0 ;          0301
18D0 ;          0302  ..CALL SCALE DATA
18D0 F829B0;    0303      LDI A.1(DTASCL); PHI ZAP
18D3 F860A0;    0304      LDI A.0(DTASCL); PLO ZAP
18D6 D0;        0305      SEP ZAP                ..ZAP = DATA SCALE
18D7 ;          0306
18D7 ;          0307  ..IN SUBROUTINE TOTAL SHIFT VALUE IS SET
18D7 ;          0308  ..ON RETURN CHECK FOR P-FFT
18D7 F8C4A808;  0309      LDI A.0(PFTFLG); PLO FRP; LDN FRP
18DB C218EA;    0310      LBZ RETNMN                ..IF S-FFT, RETURN TO MAIN
18DE ;          0311
18DE ;          0312  ..ELSE (P-FFT) GET TOTAL SHIFT VALUE
18DE F8DAAC;    0313      LDI A.0(TOTLSH); PLO ZAP1
18E1 4CBF4CAF;  0314      LDA ZAP1; PHI AC; LDA ZAP1; PLO AC
18E5 ;          0315
18E5 ;          0316  ..AND STORE IT INTO P-SHIFT LOCATION
18E5 9F5C1C;    0317      GHI AC; STR ZAP1; INC ZAP1
18E8 8F5C;      0318      GLO AC; STR ZAP1
18EA ;          0319
18EA ;          0320  RETNMN:                ..RETURN TO MAIN
18EA D5;        0321      EXIT
18EB ;          0322
18EB ;          0323
18EB ;          0324  ..#####
18EB ;          0325  ..END OF SCALE.SR

```

ASP FFT Program  
Program Code

A.2.1.3 SHAPE (Shape)

```

0000 ;          0001
0000 ;          0002 ..SHAPE.SR          20 JAN 80          5:25 PM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..          SHAPE.SR
0000 ;          0244 ..WITH
0000 ;          0245 ..          PSHAPE.SR          AND          SSHAPE.SR
0000 ;          0246 ..#####
0000 ;          0247
0000 ;          0248
0000 ;          0249 ..EXTERNAL ROUTINES
0000 ;          0250          SUBROT=#2800          ..SUBROUTINE BLOCK
0000 ;          0251          CSMULT=SUBROT+2          ..COSINE MULTIPLY
0000 ;          0252          MLTMDL=SUBROT+6          ..MIDDLE OF MULTIPLY
0000 ;          0253          SHPSET=SUBROT+14          ..SHAPE SET
0000 ;          0254
0000 ;          0255          SUBRT2=#2B50          ..SUBROUTINE BLOCK #2
0000 ;          0256          BDRYST=SUBRT2+4          ..BOUNDARY SET
0000 ;          0257          BDRYCK=SUBRT2+6          ..BOUNDARY CHECK
0000 ;          0258
0000 ;          0259 ..INTERNAL VARIABLES
0000 ;          0260
0000 ;          0261          CSPTR=CSVLU+14          ..COSINE POINTER
0000 ;          0262
0000 ;          0263 ..VECTORS TO INTERNAL ROUTINES
0000 ;          0264          ORG #1900
1900 C01906;      0265          LBR PSHAPE
1903 C01973;      0266          LBR SSHAPE
1906 ;          0267
1906 ;          0268

```

```

1906 ;      0269  ..BEGIN HERE
1906 ;      0270
1906 ;      0271  ..#####
1906 ;      0272  ..          PSHAPE.SR
1906 ;      0273  ..
1906 ;      0274  ..THIS PROGRAM MOVES DATA IN THE YPFIFO
1906 ;      0275  ..TO THE YSFIFO FOR THE FFT CALCULATION
1906 ;      0276  ..
1906 ;      0277  ..THE FIRST 75% OF THE ENTRIES ARE JUST MOVED
1906 ;      0278  ..THE LAST 25% ARE MULTIPLIED BY THE SHAPING
1906 ;      0279  ..          FUNCTION FOUND IN THE SHPTBL ARRAY
1906 ;      0280  ..#####
1906 ;      0281
1906 ;      0282  PSHAPE:          ..PSHAPE
1906 ;      0283
1906 ;      0284  ..*****
1906 ;      0285  ..          MOVE 75% OF YPFIFO DATA TO YSFIFO
1906 ;      0286  ..*****
1906 ;      0287
1906 ;      0288
1906 ;      0289  ..INITIALIZE
1906 ;      0290  ..GET PFFT POINTER, R(MA), TO P-OLDEST DATA
1906 F8A5AC; 0291          LDI A.0(YPTBL); PLO ZAP1
1909 4CBD; 0292          LDA ZAP1; PHI MA
190B 4CAD; 0293          LDA ZAP1; PLO MA
190D ;      0294  ..AND LEAVE R(ZAP1) AT YPLWA+1
190D ;      0295
190D ;      0296  ..GET R(MP) TO MOVE LENGTH VIA P-LENGTH POINTER
190D F8D2A8; 0297          LDI A.0(PLNPTR); PLO FRP
1910 48B9; 0298          LDA FRP; PHI MP
1912 48A9; 0299          LDA FRP; PLO MP
1914 ;      0300
1914 ;      0301  ..CALL BOUNDARY SET FOR BDRYOK & FREELN
1914 F82BB0; 0302          LDI A.1(BDRYST); PHI ZAP
1917 F854A0; 0303          LDI A.0(BDRYST); PLO ZAP
191A D0; 0304          SEP ZAP          ..ZAP = BOUNDARY SET
191B ;      0305
191B ;      0306  ..ON RETURN R(MP) IS AT SHAPE COUNT
191B ;      0307
191B ;      0308  ..GET R(AC) TO YSFWA
191B F83ABF; 0309          LDI A.1(YSFWA); PHI AC
191E F801AF; 0310          LDI A.0(YSFWA); PLO AC
1921 ;      0311
1921 ;      0312  MOVDTA:          ..MOVE DATA
1921 ;      0313  ..MOVE DATA FROM YPFIFO TO YSFIFO
1921 4D5F1F; 0314          LDA MA; STR AC; INC AC
1924 4D5F1F; 0315          LDA MA; STR AC; INC AC
1927 ;      0316

```



ASP FFT Program  
Program Code

```

1927 ;          0317 ..CALL BOUNDARY CHECK
1927 ;          0318 ..TO TEST FOR YPLWA+1 AND MOVE DONE
1927 D0;          0319     SEP ZAP          ..ZAP = BOUNDARY CHECK
1928 ;          0320
1928 ;          0321 ..ON RETURN CHECK D FOR DONE FLAG
1928 CA1921;      0322     LBNZ MOVDTA      ..IF NOT DONE, THEN
192B ;          0323     ..GOTO MOVE MORE DATA
192B ;          0324
192B ;          0325 ..ELSE FALL THROUGH TO SHAPE TAIL
192B ;          0326
192B ;          0327
192B ;          0328 ..*****
192B ;          0329     SHAPE 25% OF THE TAIL
192B ;          0330 ..
192B ;          0331 ..MULTIPLY THE LAST 25% OF THE P-FFT DATA
192B ;          0332 ..BY THE SHAPING FUNCTION [ -(1/2 COS) + 1/2 ]
192B ;          0333 ..GENERATED FROM THE TRIG TABLE
192B ;          0334 ..THEN STORE IT IN THE YSFIFO
192B ;          0335 ..
192B ;          0336 ..R(MA) POINTS TO THE CORRECT YPFIFO ENTRY
192B ;          0337 ..R(MQ) WILL BE THE TRIG TABLE POINTER
192B ;          0338 ..R(AC) POINTS TO THE DESTINATION
192B ;          0339 ..*****
192B ;          0340
192B ;          0341
192B ;          0342 ..INITIALIZE
192B ;          0343 ..CALL BOUNDARY SET FOR BDYRQK AND FREELN
192B F8A7AC;      0344     LDI A.0(YPTBL+2); PLO ZAP1
192E F854A0;      0345     LDI A.0(BDRYST); PLO ZAP
1931 D0;          0346     SEP ZAP          ..ZAP = BOUNDARY SET
1932 ;          0347
1932 ;          0348 ..ON RETURN R(MP) IS AT SKIP VALUE
1932 ;          0349
1932 ;          0350 ..GET R(MQ) TO COS(0)
1932 F827BE;      0351     LDI A.1(TRGLWA); PHI MQ
1935 F8FEAE;      0352     LDI A.0(TRGLWA); PLO MQ
1938 ;          0353
1938 ;          0354 GTCSVL:          ..GET COSINE VALUE
1938 ;          0355 ..GET COSINE VALUE INTO CSVLU VIA SHAPE SET
1938 F828B0;      0356     LDI A.1(SHPSET); PHI ZAP
193B F80EAO;      0357     LDI A.0(SHPSET); PLO ZAP
193E F8EDA8;      0358     LDI A.0(CSPTR+1); PLO FRP
1941 D0;          0359     SEP ZAP          ..ZAP = SHAPE SET
1942 ;          0360
1942 ;          0361 ..ON RETURN R(FRP) IS AT CSVLU.1
1942 ;          0362

```



```

1942 ; 0363 ..DO [ -(1/2 COS) + 1/2 ]
1942 ; 0364 ..GET CSVLU AND CHECK SIGN BIT
1942 48; 0365 LDA FRP
1943 FE; 0366 SHL ..CHECK SIGN
1944 CB1949; 0367 LBNF COSHLF
1947 F901; 0368 ORI #01 ..SET LOW BIT IF NEGATIVE
1949 ; 0369
1949 ; 0370 COSHLF: ..COSINE HALVE
1949 ; 0371 ..NOW SHIFT HIGH DOWN TWICE TO DIVIDE BY 2
1949 7676BA; 0372 RSHR; RSHR; PHI ZAP2
194C ; 0373
194C ; 0374 ..THEN GET AND SHIFT LOW BYTE
194C 0876; 0375 LDN FRP; SHRC
194E ; 0376
194E ; 0377 ..STORE LOW
194E 5828; 0378 STR FRP; DEC FRP
1950 ; 0379
1950 ; 0380 ..ADD 1/2 AND STORE HIGH
1950 9AFC4058; 0381 GHI ZAP2; ADI #40; STR FRP
1954 ; 0382
1954 ; 0383 ..MULTIPLY SHAPE VALUE BY YPFIFO DATA
1954 F802A0; 0384 LDI A.0(CSMULT); PLO ZAP
1957 D0; 0385 SEP ZAP ..ZAP = COSINE MULTIPLY
1958 ; 0386
1958 ; 0387 ..HALFWAY THROUGH DECREMENT CSPTR BY SKIP
1958 E9; 0388 SEX MP
1959 198EF7AE; 0389 INC MP; GLO MQ; SM; PLO MQ
195D 299E77BE; 0390 DEC MP; GHI MQ; SMB; PHI MQ
1961 ; 0391
1961 ; 0392 ..GO BACKTO STORE MULTIPLICATION RESULT INTO YSFIFO
1961 D0; 0393 SEP ZAP ..ZAP = MIDDLE OF MULTIPLY
1962 ; 0394
1962 ; 0395 ..CALL BOUNDARY CHECK
1962 ; 0396 ..TO TEST FOR YPLWA+1 AND SHAPE DONE
1962 F82BB0; 0397 LDI A.1(BDRYCK); PHI ZAP
1965 F856A0; 0398 LDI A.0(BDRYCK); PLO ZAP
1968 D0; 0399 SEP ZAP ..ZAP = BOUNDARY CHECK
1969 ; 0400
1969 ; 0401 ..ON RETURN CHECK D FOR DONE FLAG
1969 CA1938; 0402 LBNZ GTCSVL ..IF NOT DONE, THEN
196C ; 0403 ..GOTO GET NEXT COS VALUE
196C ; 0404
196C ; 0405 ..ELSE RETURN TO FFTMN
196C D5; 0406 EXIT
196D C4C4C4C4C4C4; 0407 ,#C4C4C4C4C4C4
1973 ; 0408
1973 ; 0409

```

ASP FFT Program  
Program Code

```

1973 ;      0410 ..#####
1973 ;      0411 ..          SSHAPE.SR
1973 ;      0412 ..
1973 ;      0413 ..THIS PROGRAM SHAPES THE DATA IN THE YSFIFO
1973 ;      0414 ..FOR THE FFT CALCULATION
1973 ;      0415 ..
1973 ;      0416 ..THE FIRST 12.5% AND THE LAST 25% OF THE ENTRIES
1973 ;      0417 ..ARE MULTIPLIED BY THE SHAPING FUNCTION FOUND IN
1973 ;      0418 ..THE SHPTBL ARRAY
1973 ;      0419 ..THE MIDDLE 62.5% ARE LEFT UNTOUCHED
1973 ;      0420 ..#####
1973 ;      0421 ..
1973 ;      0422 SSHAPE:          ..SSHAPE
1973 ;      0423
1973 ;      0424 ..*****
1973 ;      0425 ..          SHAPE THE FIRST 12.5%
1973 ;      0426 ..
1973 ;      0427 ..R(MA) AND R(AC) WILL POINT TO THE CORRECT
1973 ;      0428 ..          YSFIFO ENTRY
1973 ;      0429 ..R(MQ) WILL BE THE TRIG TABLE POINTER
1973 ;      0430 ..*****
1973 ;      0431 ..
1973 ;      0432 ..
1973 ;      0433 ..INITIALIZE
1973 ;      0434 ..GET SFFT POINTER, R(MA), TO S-OLDEST DATA
1973 F8AEAC; 0435             LDI A.0(YSTBL); PLO ZAP1
1976 4CBD;  0436             LDA ZAP1; PHI MA
1978 4CAD;  0437             LDA ZAP1; PLO MA
197A ;      0438 ..AND LEAVE R(ZAP1) AT YSLWA+1
197A ;      0439
197A ;      0440 ..GET R(MP) TO S-HEAD SHAPE LENGTH VIA S-LENGTH PTR
197A F8D4A8; 0441             LDI A.0(SLNPTR); PLO FRP
197D 48B9;  0442             LDA FRP; PHI MP
197F 48A9;  0443             LDA FRP; PLO MP
1981 ;      0444
1981 ;      0445 ..CALL BOUNDARY SET FOR BDRYOK & FREELN
1981 F82BB0; 0446             LDI A.1(BDRYST); PHI ZAP
1984 F854A0; 0447             LDI A.0(BDRYST); PLO ZAP
1987 D0;    0448             SEP ZAP          ..ZAP = BOUNDARY SET
1988 ;      0449
1988 ;      0450 ..ON RETURN R(MP) IS AT S-HEAD SKIP VALUE
1988 ;      0451
1988 ;      0452 ..GET R(MQ) TO COS(256)
1988 F825BE; 0453             LDI A.1(TRGLWA-512); PHI MQ
198B F8FEAE; 0454             LDI A.0(TRGLWA-512); PLO MQ
198E ;      0455

```

```

198E ;          0456 SHDSHP:                ..SFFT HEAD SHAPE
198E ;          0457 ..GET OUTPUT POINTER TO INPUT POINTER
198E 9DBF8DAF;   0458             GHI MA; PHI AC; GLO MA; PLO AC
1992 ;          0459
1992 ;          0460 ..GET COSINE VALUE INTO CSVLU VIA SHAPE SET
1992 F828B0;     0461             LDI A.1(SHPSET); PHI ZAP
1995 F80EAO;     0462             LDI A.0(SHPSET); PLO ZAP
1998 F8EDA8;     0463             LDI A.0(CSPTR+1); PLO FRP
199B D0;         0464             SEP ZAP                ..ZAP = SHAPE SET
199C ;          0465
199C ;          0466 ..ON RETURN R(FRP) IS AT CSVLU.1
199C ;          0467
199C ;          0468 ..DO [ -(1/2 COS) + 1/2 ]
199C ;          0469 ..GET CSVLU AND CHECK SIGN BIT
199C 48;         0470             LDA FRP
199D FE;         0471             SHL                ..CHECK SIGN
199E CB19A3;     0472             LBNF COSHLV
19A1 F901;       0473             ORI #01                ..SET LOW BIT IF NEGATIVE
19A3 ;          0474
19A3 ;          0475 COSHLV:                ..COSINE HALVE
19A3 ;          0476 ..NOW SHIFT HIGH DOWN TWICE TO DIVIDE BY 2
19A3 7676BA;     0477             RSHR; RSHR; PHI ZAP2
19A6 ;          0478
19A6 ;          0479 ..THEN GET AND SHIFT LOW BYTE
19A6 0876;       0480             LDN FRP; SHRC
19A8 ;          0481
19A8 ;          0482 ..STORE LOW
19A8 5828;       0483             STR FRP; DEC FRP
19AA ;          0484
19AA ;          0485 ..ADD 1/2 AND STORE HIGH
19AA 9AFC4058;   0486             GHI ZAP2; ADI #40; STR FRP
19AE ;          0487
19AE ;          0488 ..MULTIPLY SHAPE VALUE BY YPFIFO DATA
19AE F802A0;     0489             LDI A.0(CSMULT); PLO ZAP
19B1 D0;         0490             SEP ZAP                ..ZAP = COSINE MULTIPLY
19B2 ;          0491
19B2 ;          0492 ..HALFWAY THROUGH INCREMENT CSPTR BY SKIP
19B2 E9;         0493             SEX MP
19B3 198EF4AE;   0494             INC MP; GLO MQ; ADD; PLO MQ
19B7 299E74BE;   0495             DEC MP; GHI MQ; ADC; PHI MQ
19BB ;          0496
19BB ;          0497 ..GO BACKTO STORE MULTIPLICATION RESULT INTO YSFIFO
19BB D0;         0498             SEP ZAP                ..ZAP = MIDDLE OF MULTIPLY
19BC ;          0499
19BC ;          0500 ..CALL BOUNDARY CHECK
19BC ;          0501 ..TO TEST FOR YPLWA+1 AND SHAPE DONE
19BC F82BB0;     0502             LDI A.1(BDRYCK); PHI ZAP
19BF F856A0;     0503             LDI A.0(BDRYCK); PLO ZAP
19C2 D0;         0504             SEP ZAP                ..ZAP = BOUNDARY CHECK
19C3 ;          0505

```

ASP FFT Program  
Program Code

19C3 ;	0506 ..ON RETURN CHECK D FOR DONE FLAG
19C3 CA198E;	0507 LBNZ SHDSHP ..IF NOT DONE, THEN
19C6 ;	0508 ..GOTO SHAPE NEXT HEAD
19C6 ;	0509
19C6 ;	0510 ..ELSE FALL THRU TO MIDDLE SKIP
19C6 ;	0511
19C6 ;	0512
19C6 ;	0513 ..*****
19C6 ;	0514 .. SKIP THE MIDDLE 62.5%
19C6 ;	0515 ..*****
19C6 ;	0516
19C6 ;	0517
19C6 ;	0518 ..GET R(MP) TO THE MIDDLE COUNT
19C6 1919;	0519 INC MP; INC MP
19C8 ;	0520
19C8 ;	0521 ..ADD IT TO INPUT POINTER
19C8 E9;	0522 SEX MP
19C9 198DF4AD;	0523 INC MP; GLO MA; ADD; PLO MA
19CD 299D74BD;	0524 DEC MP; GHI MA; ADC; PHI MA
19D1 ;	0525
19D1 ;	0526 ..CHECK R(MA) FOR PAST YSLWA+1
19D1 8DFF01AA;	0527 GLO MA; SMI A.0(YSLWA+1); PLO ZAP2
19D5 9D7F3EBA;	0528 GHI MA; SMBI A.1(YSLWA+1); PHI ZAP2
19D9 CB19E4;	0529 LBNF STLSP ..IF PTR < LWA+1, THEN
19DC ;	0530 ..GOTO SFFT TAIL SHAPE
19DC ;	0531
19DC ;	0532 ..ELSE ADD OFFSET TO FWA
19DC 8AFC01AD;	0533 GLO ZAP2; ADI A.0(YSFWA); PLO MA
19E0 9A7C3ABD;	0534 GHI ZAP2; ADCI A.1(YSFWA); PHI MA
19E4 ;	0535
19E4 ;	0536 ..AND FALL THRU TO SFFT TAIL SHAPE
19E4 ;	0537
19E4 ;	0538



```

19E4 ;          0539 ..*****
19E4 ;          0540 ..      SHAPE THE LAST 25%
19E4 ;          0541 ..
19E4 ;          0542 ..R(MA) AND R(AC) POINT TO THE CORRECT YSFIFO ENTRY
19E4 ;          0543 ..R(MQ) WILL BE THE TRIG TABLE POINTER
19E4 ;          0544 ..*****
19E4 ;          0545
19E4 ;          0546
19E4 ;          0547 STLSP:                ..SFFT TAIL SHAPE
19E4 ;          0548 ..GET R(MP) TO TAIL SHAPE COUNT
19E4 1919;      0549      INC MP; INC MP
19E6 ;          0550
19E6 ;          0551 ..CALL BOUNDARY SET FOR BDRYOK & FREELN
19E6 F8BOAC;    0552      LDI A.0(YSTBL+2); PLO ZAP1
19E9 F854A0;    0553      LDI A.0(BDRYST); PLO ZAP
19EC D0;        0554      SEP ZAP      ..ZAP = BOUNDARY SET
19ED ;          0555
19ED ;          0556 ..ON RETURN R(MP) IS AT S-TAIL SKIP VALUE
19ED ;          0557
19ED ;          0558 ..GET R(MQ) TO COS(0)
19ED F827BE;    0559      LDI A.1(TRGLWA); PHI MQ
19F0 F8FEAE;    0560      LDI A.0(TRGLWA); PLO MQ
19F3 ;          0561
19F3 ;          0562 STLMTH:                ..SFFT TAIL SHAPE MATH
19F3 ;          0563 ..GET OUTPUT POINTER TO INPUT POINTER
19F3 9DBF8DAF;  0564      GHI MA; PHI AC; GLO MA; PLO AC
19F7 ;          0565
19F7 ;          0566 ..GET COSINE VALUE INTO CSVLU VIA SHAPE SET
19F7 F828B0;    0567      LDI A.1(SHPSET); PHI ZAP
19FA F80EAO;    0568      LDI A.0(SHPSET); PLO ZAP
19FD F8EDA8;    0569      LDI A.0(CSPTR+1); PLO FRP
1A00 D0;        0570      SEP ZAP      ..ZAP = SHAPE SET
1A01 ;          0571
1A01 ;          0572 ..ON RETURN R(FRP) IS AT CSVLU.1
1A01 ;          0573
1A01 ;          0574 ..DO [ -(1/2 COS) + 1/2 ]
1A01 ;          0575 ..GET CSVLU AND CHECK SIGN BIT
1A01 48;        0576      LDA FRP
1A02 FE;        0577      SHL      ..CHECK SIGN
1A03 CB1A08;    0578      LBNF CSHALF
1A06 F901;      0579      ORI #01      ..SET LOW BIT IF NEGATIVE
1A08 ;          0580

```



ASP FFT Program  
Program Code

1A08 ;	0581 CSHALF:	..COSINE HALVE
1A08 ;	0582 ..NOW SHIFT HIGH DOWN TWICE TO DIVIDE BY 2	
1A08 7676BA;	0583 RSHR; RSHR; PHI ZAP2	
1A0B ;	0584	
1A0B ;	0585 ..THEN GET AND SHIFT LOW BYTE	
1A0B 0876;	0586 LDN FRP; SHRC	
1A0D ;	0587	
1A0D ;	0588 ..STORE LOW	
1A0D 5828;	0589 STR FRP; DEC FRP	
1A0F ;	0590	
1A0F ;	0591 ..ADD 1/2 AND STORE HIGH	
1A0F 9AFC4058;	0592 GHI ZAP2; ADI #40; STR FRP	
1A13 ;	0593	
1A13 ;	0594 ..MULTIPLY SHAPE VALUE BY YPFIFO DATA	
1A13 F802A0;	0595 LDI A.0(CSMULT); PLO ZAP	
1A16 D0;	0596 SEP ZAP	..ZAP = COSINE MULTIPLY
1A17 ;	0597	
1A17 ;	0598 ..HALFWAY THROUGH DECREMENT CSPTR BY SKIP	
1A17 E9;	0599 SEX MP	
1A18 198EF7AE;	0600 INC MP; GLO MQ; SM; PLO MQ	
1A1C 299E77BE;	0601 DEC MP; GHI MQ; SMB; PHI MQ	
1A20 ;	0602	
1A20 ;	0603 ..GO BACKTO STORE MULTIPLICATION RESULT INTO YSFIFO	
1A20 D0;	0604 SEP ZAP	..ZAP = MIDDLE OF MULTIPLY
1A21 ;	0605	
1A21 ;	0606 ..CALL BOUNDARY CHECK	
1A21 ;	0607 ..TO TEST FOR YPLWA+1 AND SHAPE DONE	
1A21 F82BB0;	0608 LDI A.1(BDRYCK); PHI ZAP	
1A24 F856A0;	0609 LDI A.0(BDRYCK); PLO ZAP	
1A27 D0;	0610 SEP ZAP	..ZAP = BOUNDARY CHECK
1A28 ;	0611	
1A28 ;	0612 ..ON RETURN CHECK D FOR DONE FLAG	
1A28 CA19F3;	0613 LBNZ STLMTH	..IF NOT DONE, THEN
1A2B ;	0614	..GOTO SHAPE TAIL MATH
1A2B ;	0615	
1A2B ;	0616 ..ELSE RETURN TO FFTMN	
1A2B D5;	0617 EXIT	
1A2C C4C4C4C4C4C4;	0618 ,#C4C4C4C4C4C4	
1A32 ;	0619	
1A32 ;	0620	
1A32 ;	0621 ..#####	
1A32 ;	0622 ..END OF SHAPE.SR	
1A32 ;	0623	

A.2.1.4 FFT

```

0000 ;          0001
0000 ;          0002 ..FFT.SR          20 JAN 80          6:10 PM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..FFT.SR
0000 ;          0244 ..
0000 ;          0245 ..THIS PROGRAM PERFORMS THE FFT TRANSFORMATION
0000 ;          0246 ..ON THE DATA IN THE YSFIPO
0000 ;          0247 ..#####
0000 ;          0248
0000 ;          0249
0000 ;          0250 ..EXTERNAL ROUTINUES
0000 ;          0251
0000 ;          0252          SUBROT=#2800          ..SUBROUTINE BLOCK
0000 ;          0253          WRPTST=SUBROT          ..WRAPAROUND TEST
0000 ;          0254          CSMULT=SUBROT+2          ..COSINE MULTIPLY
0000 ;          0255          SNMULT=SUBROT+4          ..SINE MULTIPLY
0000 ;          0256          MLTMDL=SUBROT+6          ..MIDDLE OF MULTIPLY
0000 ;          0257          ADDSTF=SUBROT+8          ..ADD & STUFF
0000 ;          0258          NEWVAL=SUBROT+10          ..NEW VALUE
0000 ;          0259          TRGSET=SUBROT+12          ..TRIG SET
0000 ;          0260
0000 ;          0261          DTASCL=#2960          ..DATA SCALE
0000 ;          0262
0000 ;          0263 ..INTERNAL VARIABLES
0000 ;          0264 ..FFT ALGORITHM VARIABLES
0000 ;          0265
0000 ;          0266          COLPTR=CSVLU+2          ..COLUMN POINTER
0000 ;          0267
0000 ;          0268          NDBTPT=COLPTR+2          ..END BIT POINTER
0000 ;          0269          CLBTPT=NDBTPT+2          ..COLUMN BIT POINTER
0000 ;          0270          ENDBIT=CLBTPT+2          ..END BIT VALUE
0000 ;          0271          COLBIT=ENDBIT+2          ..COLUMN BIT VALUE
0000 ;          0272
0000 ;          0273          TRGCTR=COLBIT+2          ..TRIG COUNTER VALUE
0000 ;          0274          CSPTR=TRGCTR+2          ..COSINE POINTER VALUE
0000 ;          0275          SNPTR=CSPTR+2          ..SINE POINTER VALUE
0000 ;          0276          TRGPTR=SNPTR+2          ..TRIG POINTER VALUE
0000 ;          0277
0000 ;          0278          LOREAL=TRGPTR+2          ..LOW REAL ADDRESS POINTER
0000 ;          0279          LOIMAG=LOREAL+2          ..LOW IMAGINARY ADDRESS PTR
0000 ;          0280          HIREAL=LOIMAG+2          ..HIGH REAL ADDRESS POINTER
0000 ;          0281          HIIMAG=HIREAL+2          ..HIGH IMAGINARY ADDRESS PTR
0000 ;          0282
0000 ;          0283          TREAL=HIIMAG+2          ..TEMPORARY REAL VALUE
0000 ;          0284          TIMAG=TREAL+2          ..TEMPORARY IMAGINARY VALUE

```

ASP FFT Program  
Program Code

```

0000 ;          0285
0000 ;          0286
0000 ;          0287 ..*****
0000 ;          0288
0000 ;          0289          ORG #1A60
1A60 ;          0290
1A60 ;          0291 ..BEGIN HERE
1A60 ;          0292
1A60 ;          0293 FFT:          ..FAST FOURIER TRANSFORM
1A60 ;          0294
1A60 ;          0295 ..*****
1A60 ;          0296 ..          BIT/PTR SET
1A60 ;          0297 ..
1A60 ;          0298 ..FFT-LENGTH POINTER (FLNPTR) IS A POINTER TO A
1A60 ;          0299 ..          TABLE OF POINTERS
1A60 ;          0300 ..THOSE POINTERS (FLN64,FLN128,FLN256 & FLN512)
1A60 ;          0301 ..          POINT TO THE FFT-ALGORITHM DATA TABLE
1A60 ;          0302 ..          STARTING WITH THE BIT MASKS
1A60 ;          0303 ..          (0400, 0200, ..., 0040, 0000)
1A60 ;          0304 ..THIS ROUTINE SETS THE POINTERS CORRECTLY
1A60 ;          0305 ..*****
1A60 ;          0306
1A60 ;          0307
1A60 ;          0308 ..GET R(FRP) TO FLNPTR (FFT-LENGTH PTR)
1A60 F8D6;      0309          LDI A.0(FLNPTR)
1A62 A8;        0310          PLO FRP
1A63 ;          0311
1A63 ;          0312 ..LOAD FLN### POINTER VALUE INTO R(MQ)
1A63 E8;        0313          SEX FRP
1A64 72;        0314          LDXA
1A65 BE;        0315          PHI MQ
1A66 72;        0316          LDXA
1A67 AE;        0317          PLO MQ
1A68 ;          0318
1A68 ;          0319 ..GET FFT MASK ADDRESS INTO R(MP)
1A68 EE;        0320          SEX MQ
1A69 72;        0321          LDXA
1A6A B9;        0322          PHI MP
1A6B 72;        0323          LDXA
1A6C A9;        0324          PLO MP
1A6D ;          0325
1A6D ;          0326 ..GET R(ZAP1) TO CLBTPT (COLBIT POINTER)
1A6D F8E5;      0327          LDI A.0(CLBTP+1)
1A6F AC;        0328          PLO ZAP1
1A70 ;          0329
1A70 ;          0330 ..STORE FLN### POINTER INTO COLBIT POINTER
1A70 EC;        0331          SEX ZAP1
1A71 89;        0332          GLO MP
1A72 73;        0333          STXD
1A73 99;        0334          GHI MP
1A74 73;        0335          STXD

```

```

1A75 ; 0336
1A75 ; 0337 ..AND INTO NDBTPT (ENDBIT POINTER)
1A75 89; 0338 GLO MP
1A76 73; 0339 STXD
1A77 99; 0340 GHI MP
1A78 73; 0341 STXD
1A79 ; 0342
1A79 ; 0343 ..GET BIT MASK AT POINTER ADDRESS INTO R(ZAP2)
1A79 E9; 0344 SEX MP
1A7A 72; 0345 LDXA
1A7B BA; 0346 PHI ZAP2
1A7C 72; 0347 LDXA
1A7D AA; 0348 PLO ZAP2
1A7E ; 0349
1A7E ; 0350 ..GET R(ZAP1) TO COLBIT
1A7E F8E9; 0351 LDI A.0(COLBIT+1)
1A80 AC; 0352 PLO ZAP1
1A81 ; 0353
1A81 ; 0354 ..THEN STORE THE MASK INTO COLBIT
1A81 EC; 0355 SEX ZAP1
1A82 8A; 0356 GLO ZAP2
1A83 73; 0357 STXD
1A84 9A; 0358 GHI ZAP2
1A85 73; 0359 STXD
1A86 ; 0360
1A86 ; 0361 ..AND INTO ENDBIT
1A86 8A; 0362 GLO ZAP2
1A87 73; 0363 STXD
1A88 9A; 0364 GHI ZAP2
1A89 73; 0365 STXD
1A8A ; 0366
1A8A ; 0367
1A8A ; 0368

```



ASP FFT Program  
Program Code

1A8A ;	0369 ..*****
1A8A ;	0370 ..NEWCOL ..NEW COLUMN
1A8A ;	0371 ..
1A8A ;	0372 ..THIS ROUTINE INITIALIZES POINTERS, MASKS
1A8A ;	0373 .. AND COUNTERS FOR THE NEW COLUMN
1A8A ;	0374 ..AND ALSO CHECKS TO SEE IF DONE WITH THE FFT
1A8A ;	0375 ..*****
1A8A ;	0376
1A8A ;	0377 NEWCOL: ..NEW COLUMN
1A8A ;	0378
1A8A ;	0379 ..GET R(FRP) TO CLBTPT (COLBIT POINTER)
1A8A F8E4;	0380 LDI A.0(CLBTPT)
1A8C A8;	0381 PLO FRP
1A8D ;	0382
1A8D ;	0383 ..GET OLD CLBTPT ADDRESS INTO R(MP)
1A8D E8;	0384 SEX FRP
1A8E 72;	0385 LDXA
1A8F B9;	0386 PHI MP
1A90 F0;	0387 LDX
1A91 A9;	0388 PLO MP
1A92 ;	0389
1A92 ;	0390 ..INC OLD POINTER ADDR BY 2 TO NEW POINTER ADDR
1A92 19;	0391 INC MP
1A93 19;	0392 INC MP
1A94 ;	0393
1A94 ;	0394 ..STORE NEW POINTER ADDRESS BACK
1A94 89;	0395 GLO MP
1A95 73;	0396 STXD
1A96 99;	0397 GHI MP
1A97 73;	0398 STXD
1A98 ;	0399
1A98 ;	0400 ..GET VALUE AT NEW CLBTPT
1A98 ;	0401 ..IF NOT 0, SAVE IN R(ZAP2)
1A98 E9;	0402 SEX MP
1A99 72;	0403 LDXA
1A9A BA;	0404 PHI ZAP2
1A9B CA1AA5;	0405 LBNZ HIBIT
1A9E 72;	0406 LDXA
1A9F CA1AA6;	0407 LBNZ LOBIT
1AA2 ;	0408
1AA2 ;	0409 ..IF BOTH 0, YOU'RE DONE WITH THE FFT
1AA2 C01BD3;	0410 LBR DUNFFT
1AA5 ;	0411



1AA5 ;	0412 ..NOT DONE, SAVE NEW COLBIT MASK IN R(ZAP2)
1AA5 ;	0413 HIBIT: ..HI BIT NOT 0
1AA5 72;	0414 LDXA
1AA6 ;	0415 LOBIT: ..LO BIT NOT 0
1AA6 AA;	0416 PLO ZAP2
1AA7 ;	0417
1AA7 ;	0418 ..GET R(FRP) TO COLBIT
1AA7 F8E9;	0419 LDI A.0(COLBIT+1)
1AA9 A8;	0420 PLO FRP
1AAA ;	0421
1AAA ;	0422 ..STORE NEW COLBIT VALUE INTO MEMORY
1AAA E8;	0423 SEX FRP
1AAB 8A;	0424 GLO ZAP2
1AAC 73;	0425 STXD
1AAD 9A;	0426 GHI ZAP2
1AAE 73;	0427 STXD
1AAF ;	0428
1AAF ;	0429 ..GET R(FRP) TO COLPTR
1AAF F8E1;	0430 LDI A.0(COLPTR+1)
1AB1 A8;	0431 PLO FRP
1AB2 ;	0432
1AB2 ;	0433 ..GET R(ZAP1) TO TRGCTR
1AB2 F8EB;	0434 LDI A.0(TRGCTR+1)
1AB4 AC;	0435 PLO ZAP1
1AB5 ;	0436
1AB5 ;	0437 ..ZERO COLPTR
1AB5 F800;	0438 LDI #00
1AB7 73;	0439 STXD
1AB8 73;	0440 STXD
1AB9 ;	0441
1AB9 ;	0442 ..AND ZERO TRGCTR
1AB9 EC;	0443 SEX ZAP1
1ABA 73;	0444 STXD
1ABB 73;	0445 STXD
1ABC ;	0446
1ABC ;	0447 ..AND LEAVE TRGCTR VALUE, 0, IN R(MP)
1ABC B9;	0448 PHI MP
1ABD A9;	0449 PLO MP
1ABE ;	0450
1ABE ;	0451 ..CALL TRIG SET SUBROUTINE
1ABE ;	0452 ..FOR TRGPTR, SNPTR, CSPTR & CSVLU
1ABE F828B0;	0453 LDI A.1(TRGSET); PHI ZAP
1AC1 F80C;	0454 LDI A.0(TRGSET)
1AC3 A0;	0455 PLO ZAP
1AC4 D0;	0456 SEP ZAP ..ZAP = TRIG SET
1AC5 ;	0457
1AC5 ;	0458
1AC5 ;	0459

ASP FFT Program  
Program Code

```

1AC5 ;          0460 ..*****
1AC5 ;          0461 ..      MAGNITUDE CHECK
1AC5 ;          0462 ..
1AC5 ;          0463 ..THIS CHECKS THE VALUES OF THE FFT DATA
1AC5 ;          0464 ..AND SCALES THEM SO THEY WON'T OVERFLOW DURING
1AC5 ;          0465 ..THE NEW COLUMN COMPUTATION
1AC5 ;          0466 ..THAT IS: ALL VALUES FORCED BETWEEN #E000 & #1FFF
1AC5 ;          0467 ..*****
1AC5 ;          0468
1AC5 ;          0469 ..GET R(MP) TO BASE
1AC5 F830;      0470          LDI A.1(BASE)
1AC7 B9;       0471          PHI MP
1AC8 F8D8;     0472          LDI A.0(BASE)
1ACA A9;       0473          PLO MP
1ACB ;         0474
1ACB ;         0475 ..GET R(FRP) TO FFT-LENGTH
1ACB F8D0;     0476          LDI A.0(FFTLN)
1ACD A8;       0477          PLO FRP
1ACE ;         0478
1ACE ;         0479 ..GET R(ZAP1) TO YSLWA+1 POINTER
1ACE F8B0;     0480          LDI A.0(YSTBL+2)
1AD0 AC;       0481          PLO ZAP1
1AD1 ;         0482
1AD1 ;         0483 ..LOAD MAXIMUM SHIFT VALUE INTO R(ZAP2)
1AD1 ;         0484 ..THAT IS: 5 (OUT OF 7 - LEAVE 2 MSB'S CLEAR)
1AD1 F800;     0485          LDI #00
1AD3 BA;       0486          PHI ZAP2
1AD4 F805;     0487          LDI #05
1AD6 AA;       0488          PLO ZAP2
1AD7 ;         0489
1AD7 ;         0490 ..CALL DATA SCALE
1AD7 F829B0;   0491          LDI A.1(DTASCL); PHI ZAP
1ADA F860;     0492          LDI A.0(DTASCL)
1ADC A0;       0493          PLO ZAP
1ADD D0;       0494          SEP ZAP          ..ZAP = DATA SCALE
1ADE ;         0495
1ADE ;         0496
1ADE ;         0497

```

ASP FFT Program  
Program Code

```

1ADE ;      0498 ..*****
1ADE ;      0499 ..PRSKCH                ..PAIR SKIP CHECK
1ADE ;      0500 ..
1ADE ;      0501 ..THIS ROUTINUE CHECKS TO SEE IF THIS IS THE SECOND
1ADE ;      0502 ..HALF OF A PREVIOUSLY DONE PAIR BY CHECKING TO SEE
1ADE ;      0503 ..IF THE MASK BIT IS A 1
1ADE ;      0504 ..IF IT IS, IT INCREMENTS THE POINTER VALUE
1ADE ;      0505 ..BY THE MASK VALUE
1ADE ;      0506 ..*****
1ADE ;      0507
1ADE ;      0508
1ADE ;      0509 PRSKCH:                ..PAIR SKIP CHECK
1ADE ;      0510 ..GET R(FRP) TO COLPTR
1ADE F8E0;  0511             LDI A.0(COLPTR)
1AE0 A8;    0512             PLO FRP
1AE1 ;      0513
1AE1 ;      0514 ..LOAD POINTER VALUE INTO R(MP)
1AE1 E8;    0515             SEX FRP
1AE2 72;    0516             LDXA
1AE3 B9;    0517             PHI MP
1AE4 F0;    0518             LDX
1AE5 A9;    0519             PLO MP
1AE6 ;      0520 ..LEAVE R(FRP) AT COLPTR+1
1AE6 ;      0521
1AE6 ;      0522 ..GET R(ZAP1) TO COLBIT ADDRESS
1AE6 F8E8;  0523             LDI A.0(COLBIT)
1AE8 AC;    0524             PLO ZAP1
1AE9 ;      0525
1AE9 ;      0526 ..YOU WANT TO SEE IF COLPTR HAS A BIT
1AE9 ;      0527 ..             IN THE COLBIT MASK: SO
1AE9 ;      0528 ..COLPTR.AND.COLBIT
1AE9 ;      0529 ..IF NOT 0, GOTO BUMP:  PAIR DONE, DO NEXT PAIR
1AE9 ;      0530 ..IF 0,   GOTO QDADDR:  PAIR NOT DONE, CONTINUE
1AE9 ;      0531 ..             WITH COMPUTATION
1AE9 EC;    0532             SEX ZAP1
1AEA 99;    0533             GHI MP
1AEB F2;    0534             AND
1AEC 1C;    0535             INC ZAP1
1AED CA1AF5; 0536             LBNZ BUMP
1AF0 89;    0537             GLO MP
1AF1 F2;    0538             AND
1AF2 C21B24; 0539             LBZ QDADDR                ..QUAD ADDRESS SET
1AF5 ;      0540

```

ASP FFT Program  
Program Code

1AF5 ;	0541	..THE MASK PICKED A 1, BUMP THE POINTER	
1AF5 ;	0542	..BY THE VALUE OF COLBIT	
1AF5 ;	0543	..COLPTR <- COLPTR + COLBIT	
1AF5 ;	0544	BUMP:	..BUMP
1AF5 89;	0545	GLO MP	
1AF6 F4;	0546	ADD	
1AF7 A9;	0547	PLO MP	
1AF8 2C;	0548	DEC ZAP1	
1AF9 99;	0549	GHI MP	
1AFA 74;	0550	ADC	
1AFB B9;	0551	PHI MP	
1AFC ;	0552		
1AFC ;	0553	..STORE NEW COLPTR BACK	
1AFC E8;	0554	SEX FRP	..R(FRP) AT COLPTR+1
1AFD 89;	0555	GLO MP	
1AFE 73;	0556	STXD	
1AFF 99;	0557	GHI MP	
1B00 73;	0558	STXD	
1B01 ;	0559		
1B01 ;	0560	..CHECK TO SEE IF COLPTR.EQ.ENDBIT	
1B01 ;	0561	..GET R(ZAP1) TO ENDBIT	
1B01 2C;	0562	DEC ZAP1	
1B02 2C;	0563	DEC ZAP1	
1B03 ;	0564		
1B03 ;	0565	..THEN COMPARE, BRANCH TO NEWCOL IF EQUAL	
1B03 EC;	0566	SEX ZAP1	
1B04 99;	0567	GHI MP	
1B05 F3;	0568	XOR	
1B06 CALBOF;	0569	LBNZ NOTNBT	..<> 0, GOTO NOT END BIT
1B09 1C;	0570	INC ZAP1	
1BOA 89;	0571	GLO MP	
1BOB F3;	0572	XOR	
1BOC C21A8A;	0573	LBZ NEWCOL	..= 0, GOTO NEWCOLUMN
1BOF ;	0574		
1BOF ;	0575	..NOT END BIT, INCREMENT TRIG COUNTER BY 2	
1BOF ;	0576	NOTNBT:	..NOT END BIT
1BOF ;	0577		
1BOF ;	0578	..GET R(FRP) TO TRGCTR	
1BOF F8EA;	0579	LDI A.0(TRGCTR)	
1B11 A8;	0580	PLO FRP	
1B12 ;	0581		
1B12 ;	0582	..LOAD TRGCTR VALUE INTO R(MP)	
1B12 E8;	0583	SEX FRP	
1B13 72;	0584	LDXA	
1B14 B9;	0585	PHI MP	
1B15 F0;	0586	LDX	
1B16 A9;	0587	PLO MP	
1B17 ;	0588		



```

1B17 ;          0589 ..ADD 2 TO VALUE
1B17 19;        0590             INC MP
1B18 19;        0591             INC MP
1B19 ;          0592
1B19 ;          0593 ..AND STORE IT BACK
1B19 89;        0594             GLO MP
1B1A 73;        0595             STXD
1B1B 99;        0596             GHI MP
1B1C 73;        0597             STXD
1B1D ;          0598
1B1D ;          0599 ..CALL TRIG SET FOR TRGPTR, SNPTR, CSPTR & CSVLU
1B1D F828B0;    0600             LDI A.1(TRGSET); PHI ZAP
1B20 F80C;      0601             LDI A.0(TRGSET)
1B22 A0;        0602             PLO ZAP
1B23 D0;        0603             SEP ZAP             ..ZAP = TRIG SET
1B24 ;          0604
1B24 ;          0605
1B24 ;          0606
1B24 ;          0607 ..*****
1B24 ;          0608 ..QDADDR             ..QUAD ADDRESS SET
1B24 ;          0609 ..
1B24 ;          0610 ..FIRST YOU COMPUTE THE ADDRESS POINTERS OF THE
1B24 ;          0611 ..OPERANDS INTO THE FIFO, THEN CHECK FOR WRAPAROUND
1B24 ;          0612 ..
1B24 ;          0613 ..LOREAL <- BASE + COLPTR
1B24 ;          0614 ..LOIMAG <- BASE + COLPTR + 2
1B24 ;          0615 ..HIREAL <- BASE + COLPTR + COLBIT
1B24 ;          0616 ..HIIMAG <- BASE + COLPTR + COLBIT + 2
1B24 ;          0617 ..*****
1B24 ;          0618
1B24 ;          0619 QDADDR:             ..QUAD ADDRESS SET
1B24 ;          0620
1B24 ;          0621 ..GET R(FRP) TO BASE
1B24 F8D8;      0622             LDI A.0(BASE)
1B26 A8;        0623             PLO FRP
1B27 ;          0624
1B27 ;          0625 ..GET BASE VALUE INTO R(MP)
1B27 E8;        0626             SEX FRP
1B28 72;        0627             LDXA
1B29 B9;        0628             PHI MP
1B2A F0;        0629             LDX
1B2B A9;        0630             PLO MP
1B2C ;          0631
1B2C ;          0632 ..GET R(ZAP1) TO LOREAL
1B2C F8F2;      0633             LDI A.0(LOREAL)
1B2E AC;        0634             PLO ZAP1
1B2F ;          0635
1B2F ;          0636 ..GET R(FRP) TO COLPTR
1B2F F8E1;      0637             LDI A.0(COLPTR+1)
1B31 A8;        0638             PLO FRP
1B32 ;          0639

```



ASP FFT Program  
Program Code

1B32 ;	0640	..CALL ADDSTF (ADD & STUFF) FOR LOREAL & LOIMAG
1B32 F828B0;	0641	LDI A.1(ADDSTF); PHI ZAP
1B35 F808A0;	0642	LDI A.0(ADDSTF); PLO ZAP
1B38 D0;	0643	SEP ZAP ..ZAP = ADD & STUFF
1B39 ;	0644	
1B39 ;	0645	..R(MP) HAS VALUE (BASE + COLPTR)
1B39 ;	0646	..R(ZAP1) IS AT HIREAL
1B39 ;	0647	..GET R(FRP) TO COLBIT
1B39 F8E9;	0648	LDI A.0(COLBIT+1)
1B3B A8;	0649	PLO FRP
1B3C ;	0650	
1B3C ;	0651	..CALL ADDSTF (ADD & STUFF) FOR HIREAL & HIIMAG
1B3C D0;	0652	SEP ZAP ..ZAP = ADD & STUFF
1B3D ;	0653	
1B3D ;	0654	..IF IT'S A P-FFT, YOU KNOW ADDRESSES < YSLWA+1
1B3D ;	0655	..SO SKIP BOUNDARY CHECK
1B3D ;	0656	..CHECK P-FFT FLAG FOR TRUE (FF)
1B3D F8C4;	0657	LDI A.0(PFTFLG)
1B3F A8;	0658	PLO FRP
1B40 E8;	0659	SEX FRP
1B41 F0;	0660	LDX ..IF TRUE, THEN
1B42 CA1B6A;	0661	LBNZ TRGMLT ..BRANCH TO TRIG MULTIPLY
1B45 ;	0662	
1B45 ;	0663	..ELSE, IT'S A S-FFT
1B45 ;	0664	..THE ADDRESSES ARE IN LOREAL..HIIMAG
1B45 ;	0665	..CHECK FOR WRAPAROUND
1B45 ;	0666	..CHECK HIIMAG, THEN HIREAL, THEN LOIMAG,
1B45 ;	0667	..THEN LOREAL UNTIL YOU'RE UNDER THE BOUNDARY
1B45 ;	0668	
1B45 ;	0669	..GET R(ZAP) TO WRAPAROUND TEST
1B45 F800;	0670	LDI A.0(WRPTST)
1B47 A0;	0671	PLO ZAP
1B48 ;	0672	
1B48 ;	0673	..GET R(ZAP1) TO YSLWA+1 POINTER (@ YSTBL+2)
1B48 F8B0;	0674	LDI A.0(YSTBL+2)
1B4A AC;	0675	PLO ZAP1
1B4B ;	0676	
1B4B ;	0677	..GET R(FRP) TO HIIMAG POINTER
1B4B F8F8;	0678	LDI A.0(HIIMAG)
1B4D A8;	0679	PLO FRP
1B4E ;	0680	
1B4E ;	0681	..LOAD THE ADDRESS AT R(FRP) INTO R(MA)
1B4E ;	0682	LDPTR: ..LOAD POINTER
1B4E E8;	0683	SEX FRP
1B4F 72;	0684	LDXA
1B50 BD;	0685	PHI MA
1B51 F0;	0686	LDX
1B52 AD;	0687	PLO MA
1B53 ;	0688	

1B53 ;	0689	..CHECK FOR WRAPAROUND	
1B53 D0;	0690	SEP ZAP	..ZAP = WRAPAROUND TEST
1B54 ;	0691		
1B54 ;	0692	..ON RETURN, IF YOU WERE BELOW BOUNDARY	
1B54 ;	0693	..R(ZAP3.1) WILL BE #00	
1B54 9B;	0694	GHI ZAP3	..IF R(ZAP3.1) = #00
1B55 C21B6A;	0695	LBZ TRGMLT	..SKIP TO TRIG MULTIPLY
1B58 ;	0696		
1B58 ;	0697	..ELSE, STORE BACK CHANGED VALUE	
1B58 E8;	0698	SEX FRP	
1B59 8D;	0699	GLO MA	
1B5A 73;	0700	STXD	
1B5B 9D;	0701	GHI MA	
1B5C 73;	0702	STXD	
1B5D ;	0703		
1B5D ;	0704	..IF IT WAS THE LAST POINTER, SKIP TO TRIG MULTIPLY	
1B5D 88;	0705	GLO FRP	
1B5E FFF2;	0706	SMI A.0(LOREAL)	
1B60 98;	0707	GHI FRP	
1B61 7F30;	0708	SMBI A.1(LOREAL)	
1B63 CB1B6A;	0709	LBNF TRGMLT	
1B66 ;	0710		
1B66 ;	0711	..ELSE GET TO THE NEXT POINTER VALUE	
1B66 28;	0712	DEC FRP	
1B67 ;	0713		
1B67 ;	0714	..AND GO UP FOR THE NEXT CHECK	
1B67 C01B4E;	0715	LBR LDPTR	..GOTO LOAD POINTER
1B6A ;	0716		
1B6A ;	0717		
1B6A ;	0718		

ASP FFT Program  
Program Code

```

1B6A ;          0719 ..*****
1B6A ;          0720 ..TRGMLT                      ..TRIG MULTIPLY
1B6A ;          0721 ..
1B6A ;          0722 ..COMPUTE THE TEMPORARY VALUES
1B6A ;          0723 ..
1B6A ;          0724 ..TREAL <- [ M(HIREAL) * CSVLU ]
1B6A ;          0725 ..      + [ M(HIIMAG) * M(SNPTR) ]
1B6A ;          0726 ..TIMAG <- [ M(HIIMAG) * CSVLU ]
1B6A ;          0727 ..      - [ M(HIREAL) * M(SNPTR) ]
1B6A ;          0728 ..*****
1B6A ;          0729
1B6A ;          0730 TRGMLT:                      ..TRIG MULTIPLY
1B6A ;          0731
1B6A ;          0732 ..DO THE 1ST HALF OF TREAL [ M(HIREAL) * CSVLU ]
1B6A ;          0733 ..GET R(FRP) TO HIREAL POINTER
1B6A F8F6;      0734         LDI A.0(HIREAL)
1B6C A8;        0735         PLO FRP
1B6D ;          0736
1B6D ;          0737 ..GET POINTER VALUE INTO R(MA)
1B6D 48BD;      0738         LDA FRP; PHI MA
1B6F 48AD;      0739         LDA FRP; PLO MA
1B71 ;          0740 ..AND LEAVE R(FRP) POINTING TO HIIMAG
1B71 ;          0741
1B71 ;          0742 ..CALL COSINE MULTIPLY FOR THE 1ST TREAL COMPONENT
1B71 F802;      0743         LDI A.0(CSMULT)
1B73 A0;        0744         PLO ZAP
1B74 D0;        0745         SEP ZAP                      ..ZAP = COSINE MULTIPLY
1B75 ;          0746
1B75 ;          0747 ..AT HALFWAY POINT, GET R(AC) TO TREAL
1B75 F830;      0748         LDI A.1(TREAL)
1B77 BF;        0749         PHI AC
1B78 F8FA;      0750         LDI A.0(TREAL)
1B7A AF;        0751         PLO AC
1B7B ;          0752
1B7B ;          0753 ..THEN RETURN TO STORE RESULT
1B7B D0;        0754         SEP ZAP                      ..ZAP = MIDDLE OF CSMULT
1B7C ;          0755
1B7C ;          0756 ..-----
1B7C ;          0757 ..GO FOR 2ND HALF OF TIMAG [ M(HIREAL) * M(SNPTR) ]
1B7C ;          0758 ..GET R(MA) BACK TO HIREAL VALUE
1B7C 2D2D;      0759         DEC MA; DEC MA
1B7E ;          0760
1B7E ;          0761 ..GET R(ZAP1) TO SNPTR POINTER
1B7E F8EEAC;    0762         LDI A.0(SNPTR); PLO ZAP1
1B81 ;          0763
1B81 ;          0764 ..GET SNPTR VALUE INTO R(MQ)
1B81 4CBE;      0765         LDA ZAP1; PHI MQ
1B83 4CAE;      0766         LDA ZAP1; PLO MQ
1B85 ;          0767

```

```

1B85 ;          0768 ..CALL SINE MULTIPLY FOR 2ND TIMAG COMPONENT
1B85 D0;        0769             SEP ZAP             ..ZAP = SINE MULTIPLY
1B86 ;          0770             ..(LEFT THERE FROM BEFORE)
1B86 ;          0771
1B86 ;          0772 ..AT HALFWAY, R(AC) OK AT TIMAG
1B86 ;          0773 ..SO GET R(MA) TO HIIMAG VALUE FOR NEXT OPERATION
1B86 ;          0774 ..R(FRP) IS AT HIIMAG POINTER
1B86 48BD;      0775             LDA FRP; PHI MA
1B88 48AD;      0776             LDA FRP; PLO MA
1B8A ;          0777
1B8A ;          0778 ..THEN RETURN TO STORE 2ND TIMAG COMPONENT
1B8A D0;        0779             SEP ZAP             ..ZAP = MIDDLE OF SINE MULT
1B8B ;          0780
1B8B ;          0781 ..-----
1B8B ;          0782 ..NOW DO 2ND HALF OF TREAL [ M(HIIMAG) * M(SNPTR) ]
1B8B ;          0783 ..GET R(MQ) BACK TO SNPTR POINTED ADDRESS
1B8B 2E2E;      0784             DEC MQ; DEC MQ
1B8D ;          0785
1B8D ;          0786 ..RECALL SINE MULTIPLY FOR 2ND TREAL COMPONENT
1B8D D0;        0787             SEP ZAP             ..ZAP = SINE MULTIPLY
1B8E ;          0788
1B8E ;          0789 ..AT HALFWAY, GET R(AC) TO DUMMY LOCATION
1B8E ;          0790 ..(USE FREELN, A BDRYST/CK VARIABLE)
1B8E F8C6AF;    0791             LDI A.0(FREELN); PLO AC
1B91 ;          0792
1B91 ;          0793 ..GET R(MA) BACK TO HIIMAG VALUE FOR NEXT OPERATION
1B91 2D2D;      0794             DEC MA; DEC MA
1B93 ;          0795
1B93 ;          0796 ..THEN RETURN TO GET RESULT INTO R(ZAP3)
1B93 D0;        0797             SEP ZAP             ..ZAP = MIDDLE OF SINE MULT
1B94 ;          0798
1B94 ;          0799 ..GET R(AC) TO TREAL.0, AND ADD 2ND COMPONENT
1B94 F8FBFAF;   0800             LDI A.0(TREAL+1); PLO AC
1B97 EF;        0801             SEX AC
1B98 8BF473;    0802             GLO ZAP3; ADD; STXD
1B9B 9B7473;    0803             GHI ZAP3; ADC; STXD
1B9E ;          0804
1B9E ;          0805 ..-----
1B9E ;          0806 ..NOW DO 1ST HALF OF TIMAG [ M(HIMAG) * CSVLU ]
1B9E ;          0807 ..CALL COSINE MULTIPLY FOR 1ST TIIMAG COMPONENT
1B9E F802A0;    0808             LDI A.0(CSMULT); PLO ZAP
1BA1 D0;        0809             SEP ZAP             ..ZAP = COSINE MULTIPLY
1BA2 ;          0810
1BA2 ;          0811 ..AT HALFWAY GET R(AC) TO DUMMY LOCATION
1BA2 ;          0812 ..(USE FREELN, A BDRYST/CK VARIABLE)
1BA2 F8C6AF;    0813             LDI A.0(FREELN); PLO AC
1BA5 C4C4;      0814             NOP; NOP             ..AND WASTE TIME
1BA7 ;          0815
1BA7 ;          0816 ..THEN RETURN TO GET RESULT INTO R(ZAP3)
1BA7 D0;        0817             SEP ZAP             ..ZAP = MIDDLE OF CSMULT
1BA8 ;          0818

```



ASP FFT Program  
Program Code

```

1BA8 ;          0819 ..GET R(AC) TO TIMAG.0
1BA8 ;          0820 ..AND SUBTRACT MEMORY 2ND FROM R(ZAP3) 1ST
1BA8 F8FDAF;    0821          LDI A.0(TIMAG+1); PLO AC
1BAB EF;        0822          SEX AC
1BAC 8BF773;    0823          GLO ZAP3; SM; STXD
1BAF 9B7773;    0824          GHI ZAP3; SMB; STXD
1BB2 ;          0825
1BB2 ;          0826
1BB2 ;          0827
1BB2 ;          0828 ..*****
1BB2 ;          0829 ..ADD TRIG FACTORS FOR NEW QUAD
1BB2 ;          0830 ..
1BB2 ;          0831 ..NOW YOU CAN UPDATE THIS COLUMN'S QUAD. THAT IS:
1BB2 ;          0832 ..M(LOREAL) <- M(LOREAL) + TREAL
1BB2 ;          0833 ..M(LOIMAG) <- M(LOIMAG) + TIMAG
1BB2 ;          0834 ..M(HIREAL) <- M(LOREAL) - TREAL
1BB2 ;          0835 ..M(HIIMAG) <- M(LOIMAG) - TIMAG
1BB2 ;          0836 ..*****
1BB2 ;          0837
1BB2 ;          0838
1BB2 ;          0839 ..GET R(FRP) TO LOREAL POINTER
1BB2 F8F2;      0840          LDI A.0(LOREAL)
1BB4 A8;        0841          PLO FRP
1BB5 ;          0842
1BB5 ;          0843 ..GET R(ZAP1) TO TREAL VALUE
1BB5 F8FA;      0844          LDI A.0(TREAL)
1BB7 AC;        0845          PLO ZAP1
1BB8 ;          0846
1BB8 ;          0847 ..CALL NEW VALUE FOR NEW LOREAL & HIREAL
1BB8 F80A;      0848          LDI A.0(NEWVAL)
1BBA A0;        0849          PLO ZAP
1BBB D0;        0850          SEP ZAP          ..ZAP = NEW VALUE
1BBC ;          0851
1BBC ;          0852 ..GET R(FRP) TO LOIMAG POINTER
1BBC F8F4;      0853          LDI A.0(LOIMAG)
1BBE A8;        0854          PLO FRP
1BBF ;          0855
1BBF ;          0856 ..R(ZAP1) IS AT TIMAG
1BBF ;          0857 ..CALL NEW VALUE FOR LOIMAG & HIIMAG
1BBF D0;        0858          SEP ZAP          ..ZAP = NEW VALUE
1BC0 ;          0859
1BC0 ;          0860
1BC0 ;          0861

```



```

1BC0 ; 0862 ..*****
1BC0 ; 0863 ..NEXT QUAD
1BC0 ; 0864 ..
1BC0 ; 0865 ..THIS ROUTINE GETS TO THE NEXT COLUMN POINTER
1BC0 ; 0866 ..VALUE, THEN BRANCHES BACK TO CHECK THAT IT'S VALID
1BC0 ; 0867 ..*****
1BC0 ; 0868
1BC0 ; 0869
1BC0 ; 0870 ..GET R(FRP) TO COLUMN POINTER
1BC0 F8E0; 0871     LDI A.0(COLPTR)
1BC2 A8; 0872     PLO FRP
1BC3 ; 0873
1BC3 ; 0874 ..LOAD COLPTR POINTER VALUE INTO R(MP)
1BC3 E8; 0875     SEX FRP
1BC4 72; 0876     LDXA
1BC5 B9; 0877     PHI MP
1BC6 F0; 0878     LDX
1BC7 A9; 0879     PLO MP
1BC8 ; 0880
1BC8 ; 0881 ..INCREMENT THAT VALUE BY 4
1BC8 ; 0882 ..TO GET BY A PAIR OF 16 BIT WORDS
1BC8 19; 0883     INC MP
1BC9 19; 0884     INC MP
1BCA 19; 0885     INC MP
1BCB 19; 0886     INC MP
1BCC ; 0887
1BCC ; 0888 ..STORE THE NEW VALUE BACK
1BCC 89; 0889     GLO MP
1BCD 73; 0890     STXD
1BCE 99; 0891     GHI MP
1BCF 73; 0892     STXD
1BD0 ; 0893
1BD0 ; 0894 ..THEN BRANCH BACK TO CHECK FOR A VALID PAIR
1BD0 C01ADE; 0895     LBR PRSKCH     ..PAIR SKIP CHECK
1BD3 ; 0896
1BD3 ; 0897
1BD3 ; 0898
1BD3 ; 0899 ..*****
1BD3 ; 0900 ..DUNFFT     ..DONE WITH THE FFT
1BD3 ; 0901 ..*****
1BD3 ; 0902
1BD3 ; 0903 DUNFFT:     ..DONE WITH THE FFT
1BD3 D5; 0904     EXIT     ..RETURN TO CALLING PROGRAM

```

ASP FFT Program  
Program Code

A.2.1.5 UNSCRM (Unscramble)

```

0000 ;          0001
0000 ;          0002 ..UNSCRM.SR          20 JAN 80          8:35 PM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..UNSCRM.SR          UNSCRAMBLE
0000 ;          0244 ..
0000 ;          0245 ..A RESULT OF THIS DECIMATION-IN-FREQUENCY
0000 ;          0246 ..FFT ALGORITHM IS THAT THE DATA IS LEFT
0000 ;          0247 ..IN BIT-REVERSED SCRAMBLED ORDER
0000 ;          0248 ..TO UNSCRAMBLE, EXCHANGE VALUES FOR THE REAL DATA:
0000 ;          0249 ..
0000 ;          0250 ..LENGTH          SWITCHED
0000 ;          0251 .. FFT          INDEX          WITH          INDEX
0000 ;          0252 ..
0000 ;          0253 .. 512          AB CDEF GH00          HG FEDC BA00
0000 ;          0254 .. 256          OA BCDE FG00          OG FEDC BA00
0000 ;          0255 .. 128          00 ABCD EF00          00 FEDC BA00
0000 ;          0256 .. 64          00 OABC DE00          00 OEDC BA00
0000 ;          0257 ..
0000 ;          0258 ..AND FOR THE IMAGINARY DATA:
0000 ;          0259 ..
0000 ;          0260 ..LENGTH          SWITCHED
0000 ;          0261 .. FFT          INDEX          WITH          INDEX
0000 ;          0262 ..
0000 ;          0263 .. 512          AB CDEF GH10          HG FEDC BA10
0000 ;          0264 .. 256          OA BCDE FG10          OG FEDC BA10
0000 ;          0265 .. 128          00 ABCD EF10          00 FEDC BA10
0000 ;          0266 .. 64          00 OABC DE10          00 OEDC BA10
0000 ;          0267 ..#####
0000 ;          0268
0000 ;          0269
0000 ;          0270 ..EXTERNAL ROUTINUES
0000 ;          0271
0000 ;          0272          SUBROT=#2800          ..SUBROUTINUE BLOCK
0000 ;          0273          WRPTST=SUBROT          ..WRAPAROUND TEST
0000 ;          0274          ADDSTF=SUBROT+8          ..ADD & STUFF
0000 ;          0275

```

```

0000 ;      0276  ..INTERNAL VARIABLES
0000 ;      0277
0000 ;      0278      ENDCNT=CSVLU+2  ..END COUNT
0000 ;      0279      SHFCNT=ENDCNT+2  ..SHIFT COUNT
0000 ;      0280
0000 ;      0281      NMPTR=SHFCNT+2  ..NORMAL POINTER
0000 ;      0282      SWPTR=NMPTR+2  ..SWITCHED POINTER
0000 ;      0283
0000 ;      0284      NMREAL=SWPTR+2  ..NORMAL REAL INDEX
0000 ;      0285      NMIMAG=NMREAL+2  ..NORMAL IMAG INDEX
0000 ;      0286      SWREAL=NMIMAG+2  ..SWITCHED REAL INDEX
0000 ;      0287      SWIMAG=SWREAL+2  ..SWITCHED IMAG INDEX
0000 ;      0288
0000 ;      0289
0000 ;      0290      ORG #1C00
1C00 ;      0291
1C00 ;      0292  ..BEGIN HERE
1C00 ;      0293
1C00 ;      0294  UNSCRM:                      ..UNSCRAMBLE
1C00 ;      0295  ..*****
1C00 ;      0296  ..INITIALIZE
1C00 ;      0297  ..LOAD END COUNT AND SHIFT COUNT
1C00 ;      0298  ..GET R(FRP) TO FFT LENGTH POINTER
1C00 F8D6A8; 0299      LDI A.0(FLNPTR); PLO FRP
1C03 ;      0300
1C03 ;      0301  ..LOAD FLN### POINTER VALUE INTO R(MQ)
1C03 48BE48AE; 0302      LDA FRP; PHI MQ; LDA FRP; PLO MQ
1C07 ;      0303
1C07 ;      0304  ..GET END COUNT ADDRESS INTO R(MP)
1C07 4EB94EA9; 0305      LDA MQ; PHI MP; LDA MQ; PLO MP
1C0B ;      0306
1C0B ;      0307  ..GET R(FRP) TO END COUNT
1C0B F8E0A8; 0308      LDI A.0(ENDCNT); PLO FRP
1C0E ;      0309
1C0E ;      0310  ..PASS END COUNT VALUE
1C0E 495818; 0311      LDA MP; STR FRP; INC FRP
1C11 495818; 0312      LDA MP; STR FRP; INC FRP
1C14 ;      0313
1C14 ;      0314  ..PASS SHIFT COUNT VALUE
1C14 4E5818; 0315      LDA MQ; STR FRP; INC FRP
1C17 4E5818; 0316      LDA MQ; STR FRP; INC FRP
1C1A ;      0317
1C1A ;      0318  ..ZERO NORMAL POINTER
1C1A F800581858; 0319      LDI #00; STR FRP; INC FRP; STR FRP
1C1F ;      0320

```

ASP FFT Program  
Program Code

```

1C1F ;          0321 ..*****
1C1F ;          0322 GOSHFT:          ..GO SHIFT POINTERS
1C1F ;          0323 ..GET SWITCHED POINTER FROM NORMAL POINTER
1C1F ;          0324 ..VIA BITREV ROUTINE
1C1F ;          0325
1C1F ;          0326 ..GET R(FRP) TO NORMAL POINTER
1C1F F8E4A8;    0327          LDI A.0(NMPTR); PLO FRP
1C22 ;          0328
1C22 ;          0329 ..GET R(ZAP1) TO SHIFT COUNT
1C22 F8E2AC;    0330          LDI A.0(SHFCNT); PLO ZAP1
1C25 ;          0331
1C25 ;          0332 ..CALL BITREV
1C25 F81CB0;    0333          LDI A.1(BITREV); PHI ZAP
1C28 F8D5A0;    0334          LDI A.0(BITREV); PLO ZAP
1C2B D0;        0335          SEP ZAP
1C2C ;          0336
1C2C ;          0337 ..ON RETURN NORMAL POINTER IS IN R(MQ)
1C2C ;          0338 ..AND R(FRP) IS AT SWITCHED POINTER LOCATION + 1
1C2C ;          0339 ..CHECK FOR NORMAL POINTER => SWITCHED POINTER
1C2C E88EF7;    0340          SEX FRP; GLO MQ; SM
1C2F 289E77;    0341          DEC FRP; GHI MQ; SMB
1C32 C31CB1;    0342          LBDF PTRUP          ..IF NORMAL => SWITCHED
1C35 ;          0343          ..THEN GOTO POINTER UP
1C35 ;          0344          ..YOU'VE DONE THIS PAIR
1C35 ;          0345
1C35 ;          0346 ..ELSE (NEW PAIR) FALL THROUGH TO QUAD ADDRESS
1C35 ;          0347

```



```

1C35 ;      0348 ..*****
1C35 ;      0349 ..QDADDR          ..QUAD ADDRESS SET
1C35 ;      0350 ..
1C35 ;      0351 ..FIRST YOU COMPUTE THE ADDRESS POINTERS OF THE
1C35 ;      0352 ..OPERANDS INTO THE FIFO, THEN CHECK FOR WRAPAROUND
1C35 ;      0353 ..
1C35 ;      0354 ..NMREAL <- BASE + NMPTR
1C35 ;      0355 ..NMIMAG <- BASE + NMPTR + 2
1C35 ;      0356 ..SWREAL <- BASE + SWPTR
1C35 ;      0357 ..SWIMAG <- BASE + SWPTR + 2
1C35 ;      0358 ..*****
1C35 ;      0359
1C35 ;      0360 QDADDR:          ..QUAD ADDRESS SET
1C35 ;      0361
1C35 ;      0362 ..GET R(FRP) TO BASE
1C35 F8D8; 0363             LDI A.0(BASE)
1C37 A8;    0364             PLO FRP
1C38 ;      0365
1C38 ;      0366 ..GET BASE VALUE INTO R(MP)
1C38 E8;    0367             SEX FRP
1C39 72;    0368             LDXA
1C3A B9;    0369             PHI MP
1C3B F0;    0370             LDX
1C3C A9;    0371             PLO MP
1C3D ;      0372
1C3D ;      0373 ..GET R(ZAP1) TO NMREAL
1C3D F8E8; 0374             LDI A.0(NMREAL)
1C3F AC;    0375             PLO ZAP1
1C40 ;      0376
1C40 ;      0377 ..GET R(FRP) TO NMPTR
1C40 F8E5; 0378             LDI A.0(NMPTR+1)
1C42 A8;    0379             PLO FRP
1C43 ;      0380
1C43 ;      0381 ..CALL ADDSTF (ADD & STUFF) FOR NMREAL & NMIMAG
1C43 F828; 0382             LDI A.1(ADDSTF)
1C45 B0;    0383             PHI ZAP
1C46 F808; 0384             LDI A.0(ADDSTF)
1C48 A0;    0385             PLO ZAP
1C49 D0;    0386             SEP ZAP          ..ZAP = ADD & STUFF
1C4A ;      0387
1C4A ;      0388 ..GET R(FRP) TO BASE
1C4A F8D8; 0389             LDI A.0(BASE)
1C4C A8;    0390             PLO FRP
1C4D ;      0391
1C4D ;      0392 ..GET BASE VALUE INTO R(MP)
1C4D E8;    0393             SEX FRP
1C4E 72;    0394             LDXA
1C4F B9;    0395             PHI MP
1C50 F0;    0396             LDX
1C51 A9;    0397             PLO MP
1C52 ;      0398

```



ASP FFT Program  
Program Code

1C52 ;	0399	..GET R(ZAP1) TO SWREAL	
1C52 F8EC;	0400	LDI A.0(SWREAL)	
1C54 AC;	0401	PLO ZAP1	
1C55 ;	0402		
1C55 ;	0403	..GET R(FRP) TO SWPTR	
1C55 F8E7;	0404	LDI A.0(SWPTR+1)	
1C57 A8;	0405	PLO FRP	
1C58 ;	0406		
1C58 ;	0407	..CALL ADDSTF (ADD & STUFF) FOR NMREAL & NMIMAG	
1C58 D0;	0408	SEP ZAP	..ZAP = ADD & STUFF
1C59 ;	0409		
1C59 ;	0410	..IF IT'S A P-FFT, YOU KNOW ADDRESSES < YSLWA+1	
1C59 ;	0411	..SO SKIP BOUNDARY CHECK	
1C59 ;	0412	..CHECK P-FFT FLAG FOR TRUE (FF)	
1C59 F8C4;	0413	LDI A.0(PFTFLG)	
1C5B A8;	0414	PLO FRP	
1C5C E8;	0415	SEX FRP	
1C5D F0;	0416	LDX	..IF TRUE, THEN
1C5E CAlC86;	0417	LBNZ MEMSWP	..BRANCH TO MEMORY SWAP
1C61 ;	0418		
1C61 ;	0419	..ELSE, IT'S A S-FFT	
1C61 ;	0420	..THE ADDRESSES ARE IN NMREAL..SWIMAG	
1C61 ;	0421	..CHECK FOR WRAPAROUND	
1C61 ;	0422	..CHECK SWIMAG, THEN SWREAL, THEN NMIMAG,	
1C61 ;	0423	..THEN NMREAL UNTIL YOU'RE UNDER THE BOUNDARY	
1C61 ;	0424		
1C61 ;	0425	..GET R(ZAP) TO WRAPAROUND TEST	
1C61 F800;	0426	LDI A.0(WRPTST)	
1C63 A0;	0427	PLO ZAP	
1C64 ;	0428		
1C64 ;	0429	..GET R(ZAP1) TO YSLWA+1 POINTER (@ YSTBL+2)	
1C64 F8B0;	0430	LDI A.0(YSTBL+2)	
1C66 AC;	0431	PLO ZAP1	
1C67 ;	0432		
1C67 ;	0433	..GET R(FRP) TO SWIMAG POINTER	
1C67 F8EE;	0434	LDI A.0(SWIMAG)	
1C69 A8;	0435	PLO FRP	
1C6A ;	0436		
1C6A ;	0437	..LOAD THE ADDRESS AT R(FRP) INTO R(MA)	
1C6A ;	0438	LDPTR:	..LOAD POINTER
1C6A E8;	0439	SEX FRP	
1C6B 72;	0440	LDXA	
1C6C BD;	0441	PHI MA	
1C6D F0;	0442	LDX	
1C6E AD;	0443	PLO MA	
1C6F ;	0444		
1C6F ;	0445	..CHECK FOR WRAPAROUND	
1C6F D0;	0446	SEP ZAP	..ZAP = WRAPAROUND TEST
1C70 ;	0447		

```

1C70 ; 0448 ..ON RETURN, IF YOU WERE BELOW BOUNDARY
1C70 ; 0449 ..R(ZAP3.1) WILL BE #00
1C70 9B; 0450 GHI ZAP3 ..IF R(ZAP3.1) = #00
1C71 C21C86; 0451 LBZ MEMSWP ..SKIP TO MEMORY SWAP
1C74 ; 0452
1C74 ; 0453 ..ELSE, STORE BACK CHANGED VALUE
1C74 E8; 0454 SEX FRP
1C75 8D; 0455 GLO MA
1C76 73; 0456 STXD
1C77 9D; 0457 GHI MA
1C78 73; 0458 STXD
1C79 ; 0459
1C79 ; 0460 ..IF IT WAS THE LAST POINTER, SKIP TO MEMORY SWAP
1C79 88; 0461 GLO FRP
1C7A FFE8; 0462 SMI A.0(NMREAL)
1C7C 98; 0463 GHI FRP
1C7D 7F30; 0464 SMBI A.1(NMREAL)
1C7F CB1C86; 0465 LBNF MEMSWP
1C82 ; 0466
1C82 ; 0467 ..ELSE GET TO THE NEXT POINTER VALUE
1C82 28; 0468 DEC FRP
1C83 ; 0469
1C83 ; 0470 ..AND GO UP FOR THE NEXT CHECK
1C83 C01C6A; 0471 LBR LDPTR ..GOTO LOAD POINTER
1C86 ; 0472
1C86 ; 0473
1C86 ; 0474 ..*****
1C86 ; 0475 ..MEMORY SWAP
1C86 ; 0476 ..
1C86 ; 0477 ..HERE YOU EXCHANGE THE DATA
1C86 ; 0478 .. M(M(NMREAL)) WITH M(M(SWREAL))
1C86 ; 0479 .. & M(M(NMIMAG)) WITH M(M(SWIMAG))
1C86 ; 0480 ..*****
1C86 ; 0481
1C86 ; 0482 MEMSWP: ..MEMORY SWAP
1C86 ; 0483 ..GET R(FRP) TO NMREAL
1C86 F8E8A8; 0484 LDI A.0(NMREAL); PLO FRP
1C89 ; 0485
1C89 ; 0486 ..GET R(ZAP1) TO SWREAL
1C89 F8ECAC; 0487 LDI A.0(SWREAL); PLO ZAP1
1C8C ; 0488
1C8C ; 0489 ..SET FIRST PASS FLAG IN R(ZAP3.H)
1C8C F800BB; 0490 LDI #00; PHI ZAP3
1C8F ; 0491

```

ASP FFT Program  
Program Code

1C8F ;	0492 SWAP:	..SWAP MEMORY
1C8F ;	0493 ..LOAD NM-----	POINTER INTO R(MP)
1C8F 48B948A9;	0494	LDA FRP; PHI MP; LDA FRP; PLO MP
1C93 ;	0495 ..AND LEAVE R(FRP) AT NM-----	PTR + 2
1C93 ;	0496	
1C93 ;	0497 ..LOAD SW-----	POINTER INTO R(MA)
1C93 4CBD4CAD;	0498	LDA ZAP1; PHI MA; LDA ZAP1; PLO MA
1C97 ;	0499 ..AND LEAVE R(ZAP1) AT SW-----	PTR + 2
1C97 ;	0500	
1C97 ;	0501 ..LOAD NORMAL VALUE	
1C97 49BE09AE;	0502	LDA MP; PHI MQ; LDN MP; PLO MQ
1C9B ;	0503	
1C9B ;	0504 ..LOAD SWITCHED VALUE	
1C9B 4DBF0D;	0505	LDA MA; PHI AC; LDN MA
1C9E ;	0506	
1C9E ;	0507 ..STORE SWITCHED VALUE	
1C9E E9739F73;	0508	SEX MP; STXD; GHI AC; STXD
1CA2 ;	0509	
1CA2 ;	0510 ..STORE NORMAL VALUE	
1CA2 ED8E739E73;	0511	SEX MA; GLO MQ; STXD; GHI MQ; STXD
1CA7 ;	0512	
1CA7 ;	0513 ..IF SECOND PASS, THEN JUMP OUT	
1CA7 9BCA1CB1;	0514	GHI ZAP3; LBNZ PTRUP
1CAB ;	0515	
1CAB ;	0516 ..ELSE WAS FIRST, SET SECOND PASS FLAG	
1CAB F8FFB8;	0517	LDI #FF; PHI ZAP3
1CAE ;	0518	
1CAE ;	0519 ..AND GO BACK FOR SECOND	
1CAE C01C8F;	0520	LBR SWAP
1CB1 ;	0521	
1CB1 ;	0522 ..*****	
1CB1 ;	0523 PTRUP:	..POINTER UP
1CB1 ;	0524 ..THIS ROUTINE INCREMENTS THE NORMAL POINTER BY 4	
1CB1 ;	0525 ..THEN CHECKS FOR => END COUNT (DONE CONDITION)	
1CB1 ;	0526	
1CB1 ;	0527 ..GET R(FRP) TO NORMAL POINTER	
1CB1 F8E4A8;	0528	LDI A.0(NMPTR); PLO FRP
1CB4 ;	0529	
1CB4 ;	0530 ..LOAD VALUE INTO R(MP)	
1CB4 E872B9FOA9;	0531	SEX FRP; LDXA; PHI MP; LDX; PLO MP
1CB9 ;	0532	
1CB9 ;	0533 ..INCREMENT BY 4	
1CB9 19191919;	0534	INC MP; INC MP; INC MP; INC MP
1CBD ;	0535	
1CBD ;	0536 ..STORE BACK	
1CBD 89739958;	0537	GLO MP; STXD; GHI MP; STR FRP
1CC1 ;	0538	
1CC1 ;	0539 ..GET R(ZAP1) TO END COUNT	
1CC1 F8E1AC;	0540	LDI A.0(ENDCNT+1); PLO ZAP1
1CC4 ;	0541	



```

1CC4 ;          0542 ..CHECK FOR POINTER < COUNT
1CC4 EC89F7;    0543         SEX ZAP1; GLO MP; SM
1CC7 2C9977;    0544         DEC ZAP1; GHI MP; SMB
1CCA CB1C1F;    0545         LBNF GOSHFT         ..IF POINTER < COUNT
1CCD ;          0546         ..THEN GOTO GO SHIFT
1CCD ;          0547
1CCD ;          0548 ..ELSE DONE, RETURN TO MAIN
1CCD D5;        0549         EXIT
1CCE C4C4C4C4C4C4; 0550         ,#C4C4C4C4C4C4
1CD4 ;          0551
1CD4 ;          0552 ..#####
1CD4 ;          0553 ..BITREV                     BIT REVERSAL
1CD4 ;          0554 ..
1CD4 ;          0555 ..THIS ROUTINE SHIFTS NORMAL POINTER OFF THE RIGHT
1CD4 ;          0556 ..AND THEN LEFT INTO SWITCHED POINTER
1CD4 ;          0557 ..SHIFT COUNT TIMES
1CD4 ;          0558 ..
1CD4 ;          0559 ..IT ASSUMES R(FRP) IS AT NORMAL POINTER
1CD4 ;          0560 ..AND R(ZAP1) IS AT SHIFT COUNT
1CD4 ;          0561 ..
1CD4 ;          0562 ..ON RETURN SWITCHED POINTER IS IN R(ZAP3)
1CD4 ;          0563 ..         AND AT M(R(FRP+2))
1CD4 ;          0564 ..#####
1CD4 ;          0565
1CD4 ;          0566 ..RETURN PORTION
1CD4 ;          0567 EXBTRV:                     ..EXIT BIT REVERSAL
1CD4 D3;        0568         SEP PC                     ..RESET TO R(PC)
1CD5 ;          0569
1CD5 ;          0570 ..ENTER HERE
1CD5 ;          0571 BITREV:                     ..BIT REVERSAL
1CD5 ;          0572 ..LOAD NORMAL POINTER INTO R(MP) & R(MQ)
1CD5 48B9BE;    0573         LDA FRP; PHI MP; PHI MQ
1CD8 48A9AE;    0574         LDA FRP; PLO MP; PLO MQ
1CDB ;          0575 ..AND LEAVE R(FRP) AT SWITCHED POINTER
1CDB ;          0576
1CDB ;          0577 ..LOAD SHIFT COUNT INTO R(ZAP2)
1CDB 4CBA4CAA;  0578         LDA ZAP1; PHI ZAP2; LDA ZAP1; PLO ZAP2
1CDF ;          0579
1CDF ;          0580 ..USE R(ZAP3) FOR SWITCHED POINTER, CLEAR IT
1CDF F800BBAB;  0581         LDI #00; PHI ZAP3; PLO ZAP3
1CE3 ;          0582
1CE3 ;          0583 ..SHIFT NORMAL POINTER INTO SWITCHED POINTER THRU DF
1CE3 ;          0584 SHFARO:                     ..SHIFT AROUND
1CE3 99F6B9;    0585         GHI MP; SHR; PHI MP
1CE6 8976A9;    0586         GLO MP; SHRC; PLO MP
1CE9 ;          0587
1CE9 8B7EAB;    0588         GLO ZAP3; SHLC; PLO ZAP3
1CEC 9B7EBB;    0589         GHI ZAP3; SHLC; PHI ZAP3
1CEF ;          0590

```

ASP FFT Program  
Program Code

1CEF ;	0591 ..COUNT DOWN VIA R(ZAP2)
1CEF 2A8A;	0592 DEC ZAP2; GLO ZAP2
1CF1 CA1CE3;	0593 LBNZ SHFARO ..IF NOT #00
1CF4 ;	0594 ..SHIFT AROUND AGAIN
1CF4 ;	0595
1CF4 ;	0596 ..ELSE STORE SWITCHED POINTER VALUE
1CF4 9B5818;	0597 GHI ZAP3; STR FRP; INC FRP
1CF7 8B58;	0598 GLO ZAP3; STR FRP
1CF9 ;	0599 ..AND LEAVE R(FRP) AT SWITCHED POINTER + 1
1CF9 ;	0600
1CF9 ;	0601 ..THEN RETURN THRU TOP
1CF9 C01CD4;	0602 LBR EXBTRV
1CFC ;	0603
1CFC ;	0604 ..#####
1CFC ;	0605 ..END OF UNSCRM.SR



A.2.1.6 ORDER

```

0000 ;          0001
0000 ;          0002 ..ORDER.SR          24 MAY 80          8:45 PM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..ORDER.SR          COLUMN REORDER
0000 ;          0244 ..
0000 ;          0245 ..A RESULT OF SPLITTING THE 2M FFT DATA INTO TWO M
0000 ;          0246 ..FUNCTIONS (REAL & IMAGINARY) AND THEN TRANSFORMING
0000 ;          0247 ..THE M SERIES IS THAT OUR OUTPUT IS ORDERED AS
0000 ;          0248 ..PER THE SPLIT SERIES, NOT THE ORIGINAL.
0000 ;          0249 ..TO REGAIN THE ORIGINAL, REORDER AS FOLLOWS:
0000 ;          0250 ..(SEE "THE FAST FOURIER TRANSFORM" BRIGHAM P.169)
0000 ;          0251 ..
0000 ;          0252 ..FOR N = [0...M-1]
0000 ;          0253 ..2 * XR(N) = [R(N)+R(M-N)]
0000 ;          0254 ..      + COS(2*PI*N/2*M) * [I(N)+I(M-N)]
0000 ;          0255 ..      - SIN(2*PI*N/2*M) * [R(N)-R(M-N)]
0000 ;          0256 ..2 * XI(N) = [I(N)-I(M-N)]
0000 ;          0257 ..      - SIN(2*PI*N/2*M) * [I(N)+I(M-N)]
0000 ;          0258 ..      - COS(2*PI*N/2*M) * [R(N)-R(M-N)]
0000 ;          0259 ..
0000 ;          0260 ..FOR N = [1...M-1]
0000 ;          0261 ..2 * XR(M-N) = * [R(M-N)+R(N)]
0000 ;          0262 ..      + COS(2*PI*(M-N)/2*M) * [I(M-N)+I(N)]
0000 ;          0263 ..      - SIN(2*PI*(M-N)/2*M) * [R(M-N)-R(N)]
0000 ;          0264 ..2 * XI(M-N) = [I(M-N)-I(N)]
0000 ;          0265 ..      - SIN(2*PI*(M-N)/2*M) * [I(M-N)+I(N)]
0000 ;          0266 ..      - COS(2*PI*(M-N)/2*M) * [R(M-N)-R(N)]
0000 ;          0267 ..
0000 ;          0268 ..WE WILL USE THE FACTS:
0000 ;          0269 ..
0000 ;          0270 .. SIN(2*PI*(M-N)/2*M) = + SIN(2*PI*N/2*M)
0000 ;          0271 .. COS(2*PI*(M-N)/2*M) = - COS(2*PI*N/2*M)
0000 ;          0272 ..AND [R(M-N)-R(N)] = - [R(N)-R(M-N)]
0000 ;          0273 ..
0000 ;          0274 ..AND ALLOW THE RESULTS TO BE SCALED BY 2
0000 ;          0275 ..SO:
0000 ;          0276 ..

```

ASP FFT Program  
Program Code

```

0000 ;      0277 ..LOREAL = RLADD + (LOCOS*IMADD) - (LOSIN*RLSBLH)
0000 ;      0278 ..LOIMAG = IMSBLH - (LOSIN*IMADD) - (LOCOS*RLSBLH)
0000 ;      0279 ..HIREAL = RLADD - (LOCOS*IMADD) + (LOSIN*RLSBLH)
0000 ;      0280 ..HIIMAG = IMSBHL - (LOSIN*IMADD) - (LOCOS*RLSBLH)
0000 ;      0281 ..
0000 ;      0282 ..WHERE: RLADD = (LOREAL+HIREAL)
0000 ;      0283 ..      RLSBLH = (LOREAL-HIREAL)
0000 ;      0284 ..      RLSBHL = (HIREAL-LOREAL)
0000 ;      0285 ..      IMADD = (LOIMAG+HIIMAG)
0000 ;      0286 ..      IMSBLH = (LOIMAG-HIIMAG)
0000 ;      0287 ..      IMSBHL = (HIIMAG-LOIMAG)
0000 ;      0288 ..#####
0000 ;      0289
0000 ;      0290
0000 ;      0291 ..EXTERNAL ROUTINES
0000 ;      0292
0000 ;      0293          SUBROT=#2800      ..SUBROUTINE BLOCK
0000 ;      0294          WRPTST=SUBROT  ..WRAPAROUND TEST
0000 ;      0295          DTASCL=#2960   ..DATA SCALE
0000 ;      0296
0000 ;      0297 ..INTERNAL VARIABLES
0000 ;      0298
0000 ;      0299          LOREAL=CSVLU+2  ..LOW REAL ADDRESS POINTER
0000 ;      0300          LOIMAG=LOREAL+2 ..LOW IMAGINARY ADDRESS PTR
0000 ;      0301          HIREAL=LOIMAG+2 ..HIGH REAL ADDRESS POINTER
0000 ;      0302          HIIMAG=HIREAL+2 ..HIGH IMAGINARY ADDRESS PTR
0000 ;      0303
0000 ;      0304          TRGINC=HIIMAG+2 ..TRIG INCREMENT
0000 ;      0305          LOSIN=TRGINC+2  ..LOW SINE VALUE
0000 ;      0306          LOCOS=LOSIN+2  ..LOW COSINE VALUE
0000 ;      0307
0000 ;      0308          RLADD=LOCOS+2     ..REAL ADD
0000 ;      0309          RLSBLH=RLADD+2  ..REAL SUB LOW - HIGH
0000 ;      0310          RLSBHL=RLSBLH+2  ..REAL SUB HIGH - LOW
0000 ;      0311          IMADD=RLSBHL+2  ..IMAG ADD
0000 ;      0312          IMSBLH=IMADD+2   ..IMAG SUB LOW - HIGH
0000 ;      0313          IMSBHL=IMSBLH+2  ..IMAG SUB HIGH - LOW
0000 ;      0314
0000 ;      0315
0000 ;      0316          ORG #1D20
0000 ;      0317
0000 ;      0318 ..BEGIN HERE
0000 ;      0319

```

```

1D20 ;          0320 ORDER:                ..COLUMN REORDER
1D20 ;          0321 ..*****
1D20 ;          0322 ..INITIALIZE
1D20 ;          0323 ..GET R(FRP) TO FFT LENGTH POINTER
1D20 F8D6A8;    0324         LDI A.0(FLNPTR); PLO FRP
1D23 ;          0325
1D23 ;          0326 ..LOAD FLN### POINTER VALUE INTO R(MP)
1D23 48B948A9; 0327         LDA FRP; PHI MP; LDA FRP; PLO MP
1D27 ;          0328
1D27 ;          0329 ..GET R(MP) TO THIS TRIG INC VALUE
1D27 19191919; 0330         INC MP; INC MP; INC MP; INC MP
1D2B ;          0331
1D2B ;          0332 ..GET R(ZAP1) TO TRIG INC VARIABLE LOCATION
1D2B F8E8AC;    0333         LDI A.0(TRGINC); PLO ZAP1
1D2E ;          0334
1D2E ;          0335 ..PASS TRIG INC VALUE
1D2E 49BB5C1C; 0336         LDA MP; PHI ZAP3; STR ZAP1; INC ZAP1
1D32 49AB5C;    0337         LDA MP; PLO ZAP3; STR ZAP1
1D35 ;          0338
1D35 ;          0339 ..GET R(ZAP1) TO LOCOS.0
1D35 F8EDAC;    0340         LDI A.0(LOCOS+1); PLO ZAP1
1D38 ;          0341
1D38 ;          0342 ..LOCOS <- TRGLWA - TRGINC
1D38 EC;        0343         SEX ZAP1
1D39 8BFDFF73; 0344         GLO ZAP3; SDI A.0(TRGLWA); STXD
1D3D 9B7D2773; 0345         GHI ZAP3; SDBI A.1(TRGLWA); STXD
1D41 ;          0346
1D41 ;          0347 ..LOSIN <- TRGFWA + TRGINC
1D41 8BFCFE73; 0348         GLO ZAP3; ADI A.0(TRGFWA); STXD
1D45 9B7C2673; 0349         GHI ZAP3; ADCI A.1(TRGFWA); STXD
1D49 ;          0350 ..*****
1D49 ;          0351 ..SCALE FFT DATA TO MAXIMUM #1FFF
1D49 ;          0352 ..(SO NO OVERFLOWS)
1D49 ;          0353
1D49 ;          0354 ..GET R(MP) TO BASE
1D49 F830B9;    0355         LDI A.1(BASE); PHI MP
1D4C F8D8A9;    0356         LDI A.0(BASE); PLO MP
1D4F ;          0357
1D4F ;          0358 ..GET R(FRP) TO FFT LENGTH
1D4F F8D0A8;    0359         LDI A.0(FFTLN); PLO FRP
1D52 ;          0360
1D52 ;          0361 ..GET R(ZAP1) TO YSLWA+1 POINTER
1D52 F8B0AC;    0362         LDI A.0(YSTBL+2); PLO ZAP1
1D55 ;          0363
1D55 ;          0364 ..LOAD MAXIMUM SHIFT (5) INTO R(ZAP2)
1D55 F800BAF805AA; 0365         LDI #00; PHI ZAP2; LDI #05; PLO ZAP2
1D5B ;          0366

```



ASP FFT Program  
Program Code

```

1D5B ;          0367 ..CALL DATA SCALE
1D5B F829B0;    0368         LDI A.1(DTASCL); PHI ZAP
1D5E F860A0;    0369         LDI A.0(DTASCL); PLO ZAP
1D61 D0;        0370         SEP ZAP          ..ZAP = DATA SCALE
1D62 ;          0371
1D62 ;          0372
1D62 ;          0373 ..*****
1D62 ;          0374 ..SET LOREAL <- BASE
1D62 ;          0375 ..      LOIMAG <- BASE + 2
1D62 ;          0376 ..      HIREAL <- BASE + FFTLN - 4
1D62 ;          0377 ..      HIIMAG <- BASE + FFTLN - 2
1D62 ;          0378 ..AND CHECK FOR WRAPAROUND
1D62 ;          0379
1D62 ;          0380 ..LOAD BASE ADDRESS VALUE INTO R(MA)
1D62 F8D8A8;    0381         LDI A.0(BASE); PLO FRP
1D65 48BD48AD;  0382         LDA FRP; PHI MA; LDA FRP; PLO MA
1D69 ;          0383
1D69 ;          0384 ..STORE BASE + 2 INTO LOIMAG
1D69 F8E3A8;    0385         LDI A.0(LOIMAG+1); PLO FRP
1D6C E8;        0386         SEX FRP
1D6D 8DFC0273;  0387         GLO MA; ADI #02; STXD
1D71 9D7C0073;  0388         GHI MA; ADCI #00; STXD
1D75 ;          0389
1D75 ;          0390 ..STORE BASE ADDRESS VALUE INTO LOREAL
1D75 8D739D73;  0391         GLO MA; STXD; GHI MA; STXD
1D79 ;          0392
1D79 ;          0393 ..ADD FFTLN VALUE TO BASE ADDRESS VALUE
1D79 ;          0394 ..AND SAVE IN R(MA)
1D79 F8D1A8;    0395         LDI A.0(FFTLN+1); PLO FRP
1D7C 8DF4AD28;  0396         GLO MA; ADD; PLO MA; DEC FRP
1D80 9D74BD;    0397         GHI MA; ADC; PHI MA
1D83 ;          0398
1D83 ;          0399 ..STORE (BASE + FFTLN) - 2 INTO HIIMAG
1D83 F8E7A8;    0400         LDI A.0(HIIMAG+1); PLO FRP
1D86 8DFF0273;  0401         GLO MA; SMI #02; STXD
1D8A 9D7F0073;  0402         GHI MA; SMBI #00; STXD
1D8E ;          0403
1D8E ;          0404 ..STORE (BASE + FFTLN) - 4 INTO HIREAL
1D8E 8DFF0473;  0405         GLO MA; SMI #04; STXD
1D92 9D7F0073;  0406         GHI MA; SMBI #00; STXD
1D96 ;          0407
1D96 ;          0408 ..CHECK FOR WRAPAROUND
1D96 ;          0409 ..CHECK HIIMAG, THEN HIREAL, THEN LOIMAG
1D96 ;          0410 ..UNTIL YOU'RE UNDER BOUNDARY
1D96 ;          0411
1D96 ;          0412 ..GET R(ZAP) TO WRAPAROUND TEST
1D96 F828B0;    0413         LDI A.1(WRPTST); PHI ZAP
1D99 F800A0;    0414         LDI A.0(WRPTST); PLO ZAP
1D9C ;          0415

```

```

1D9C ;          0416 ..GET R(ZAP1) TO YSLWA+1 POINTER (@ YSTBL+2)
1D9C F8B0AC;    0417             LDI A.0(YSTBL+2); PLO ZAP1
1D9F ;          0418
1D9F ;          0419 ..GET R(FRP) TO HIIMAG POINTER
1D9F F8E6A8;    0420             LDI A.0(HIIMAG); PLO FRP
1DA2 ;          0421
1DA2 ;          0422 ..LOAD THE ADDRESS AT R(FRP) INTO R(MA)
1DA2 ;          0423 LDPTR:             ..LOAD POINTER
1DA2 E8;        0424             SEX FRP
1DA3 72BDF0AD;  0425             LDXA; PHI MA; LDX; PLO MA
1DA7 ;          0426
1DA7 ;          0427 ..CHECK FOR WRAPAROUND
1DA7 D0;        0428             SEP ZAP             ..ZAP = WRAPAROUND TEST
1DA8 ;          0429
1DA8 ;          0430 ..ON RETURN, IF YOU WERE BELOW BOUNDARY (OK)
1DA8 ;          0431 ..R(ZAP3.1) WILL BE #00
1DA8 9B;        0432             GHI ZAP3             ..IF R(ZAP3.1) = #00
1DA9 C21DBE;    0433             LBZ CMPZRO             ..SKIP TO COMPUTE ZERO
1DAC ;          0434
1DAC ;          0435 ..ELSE STORE BACK CHANGED VALUE
1DAC E8;        0436             SEX FRP
1DAD 8D739D73;  0437             GLO MA; STXD; GHI MA; STXD
1DB1 ;          0438
1DB1 ;          0439 ..IF IT WAS THE LAST POINTER, SKIP TO COMPUTE ZERO
1DB1 88FFE2;    0440             GLO FRP; SMI A.0(LOIMAG)
1DB4 987F30;    0441             GHI FRP; SMBI A.1(LOIMAG)
1DB7 CB1DBE;    0442             LBNF CMPZRO             ..IF R(FRP) < LOIMAG, THEN
1DBA ;          0443             ..SKIP TO COMPUTE ZERO
1DBA ;          0444
1DBA ;          0445 ..ELSE GET TO THE NEXT POINTER
1DBA 28;        0446             DEC FRP
1DBB ;          0447
1DBB ;          0448 ..AND GO BACK FOR THE NEXT CHECK
1DBB C01DA2;    0449             LBR LDPTR             ..GOTO LOAD POINTER
1DBE ;          0450
1DBE ;          0451
1DBE ;          0452 ..*****
1DBE ;          0453 ..COMPUTE LOREAL(0) <- 2 * (LOREAL + LOIMAG)
1DBE ;          0454 ..      AND LOIMAG(0) <- #0000 AND STORE
1DBE ;          0455
1DBE ;          0456 CMPZRO:             ..COMPUTE ZERO VALUES
1DBE ;          0457 ..GET R(MA) TO LOREAL(0)
1DBE F8E0A8;    0458             LDI A.0(LOREAL); PLO FRP
1DC1 48BD48AD;  0459             LDA FRP; PHI MA; LDA FRP; PLO MA
1DC5 ;          0460
1DC5 ;          0461 ..GET R(MQ) TO LOIMAG(0)
1DC5 48BE08AE;  0462             LDA FRP; PHI MQ; LDN FRP; PLO MQ
1DC9 ;          0463

```



ASP FFT Program  
Program Code

```

1DC9 ; 0464 ..ADD LOREAL(0) + LOIMAG(0)
1DC9 4EBFOEAF; 0465 LDA MQ; PHI AC; LDN MQ; PLO AC
1DCD ED; 0466 SEX MA
1DCE 1D8FF4AF; 0467 INC MA; GLO AC; ADD; PLO AC
1DD2 2D9F74BF; 0468 DEC MA; GHI AC; ADC; PHI AC
1DD6 ; 0469
1DD6 ; 0470 ..THEN MULTIPLY BY 2 AND STORE AT LOREAL(0) ADDRESS
1DD6 1D; 0471 INC MA
1DD7 8FFE73; 0472 GLO AC; SHL; STXD
1DDA 9F7E73; 0473 GHI AC; SHLC; STXD
1DDD ; 0474
1DDD ; 0475 ..THEN STORE #0000 AT LOIMAG ADDRESS
1DDD F8005E2E5E; 0476 LDI #00; STR MQ; DEC MQ; STR MQ
1DE2 ; 0477
1DE2 ; 0478 ..THEN GOTO INCREMENT LO----- POINTERS
1DE2 ; 0479 ..(R(FRP) IS OK AT LOIMAG.0, JUST SEX FRP)
1DE2 E8; 0480 SEX FRP
1DE3 C01EFA; 0481 LBR LOUP
1DE6 ; 0482
1DE6 ; 0483
1DE6 ; 0484 ..*****
1DE6 ; 0485 MIDSIX: ..COMPUTE 6 MIDDLE VALUES
1DE6 ; 0486
1DE6 ; 0487 ..GET R(FRP) TO LOREAL POINTER
1DE6 F8E0A8; 0488 LDI A.0(LOREAL); PLO FRP
1DE9 ; 0489
1DE9 ; 0490 ..GET R(ZAP1) TO HIREAL POINTER
1DE9 F8E4AC; 0491 LDI A.0(HIREAL); PLO ZAP1
1DEC ; 0492
1DEC ; 0493 ..GET R(ZAP3) TO RLSBHL+1 ADDRESS
1DEC F830BB; 0494 LDI A.1(RLSBHL+1); PHI ZAP3
1DEF F8F3AB; 0495 LDI A.0(RLSBHL+1); PLO ZAP3
1DF2 ; 0496
1DF2 ; 0497 ..CALL COMBO
1DF2 ; 0498 ..TO GET RLADD, RLSBLH & RLSBHL
1DF2 F81FBO; 0499 LDI A.1(COMBO); PHI ZAP
1DF5 F882A0; 0500 LDI A.0(COMBO); PLO ZAP
1DF8 D0; 0501 SEP ZAP ..ZAP = COMBO
1DF9 ; 0502
1DF9 ; 0503 ..ON RETURN
1DF9 ; 0504 ..R(FRP) IS AT LOIMAG POINTER
1DF9 ; 0505 ..R(ZAP1) IS AT HIIMAG POINTER
1DF9 ; 0506 ..GET R(ZAP3) TO IMSBHL+1 ADDRESS
1DF9 F8F9AB; 0507 LDI A.0(IMSBHL+1); PLO ZAP3
1DFC ; 0508
1DFC ; 0509 ..RECALL COMBO
1DFC ; 0510 ..FOR IMADD, IMSBLH & IMSBHL
1DFC D0; 0511 SEP ZAP ..ZAP = COMBO
1DFD ; 0512

```

1DFD ;	0513	..*****
1DFD ;	0514	..STORE THE FINAL VALUES
1DFD ;	0515	..DO THE FIRST COLUMN STORES
1DFD ;	0516	
1DFD ;	0517	..GET R(FRP) TO LOREAL POINTER
1DFD F8EOA8;	0518	LDI A.0(LOREAL); PLO FRP
1EO0 ;	0519	
1EO0 ;	0520	..LOAD LOREAL ADDRESS INTO R(MP)
1EO0 48B948A9;	0521	LDA FRP; PHI MP; LDA FRP; PLO MP
1EO4 ;	0522	
1EO4 ;	0523	..GET R(ZAP1) TO RLADD
1EO4 F8EEAC;	0524	LDI A.0(RLADD); PLO ZAP1
1EO7 ;	0525	
1EO7 ;	0526	..PASS RLADD TO M(LOREAL)
1EO7 4CBF5919;	0527	LDA ZAP1; PHI AC; STR MP; INC MP
1EOB 4CAF59;	0528	LDA ZAP1; PLO AC; STR MP
1EOE ;	0529	
1EOE ;	0530	..-----
1EOE ;	0531	..LOAD LOIMAG ADDRESS INTO R(MP)
1EOE 48B948A9;	0532	LDA FRP; PHI MP; LDA FRP; PLO MP
1E12 ;	0533	
1E12 ;	0534	..GET R(ZAP1) TO IMSBLH
1E12 F8F6AC;	0535	LDI A.0(IMSBLH); PLO ZAP1
1E15 ;	0536	
1E15 ;	0537	..PASS IMSBLH TO M(LOIMAG)
1E15 4C5919;	0538	LDA ZAP1; STR MP; INC MP
1E18 4C59;	0539	LDA ZAP1; STR MP
1E1A ;	0540	
1E1A ;	0541	..-----
1E1A ;	0542	..LOAD HIREAL ADDRESS INTO R(MP)
1E1A 48B948A9;	0543	LDA FRP; PHI MP; LDA FRP; PLO MP
1E1E ;	0544	
1E1E ;	0545	..STORE RLADD INTO M(HIREAL)
1E1E 9F5919;	0546	GHI AC; STR MP; INC MP
1E21 8F59;	0547	GLO AC; STR MP
1E23 ;	0548	
1E23 ;	0549	..-----
1E23 ;	0550	..LOAD HIIMAG ADDRESS INTO R(MP)
1E23 48B948A9;	0551	LDA FRP; PHI MP; LDA FRP; PLO MP
1E27 ;	0552	
1E27 ;	0553	..GET R(ZAP1) TO IMSBHL
1E27 F8F8AC;	0554	LDI A.0(IMSBHL); PLO ZAP1
1E2A ;	0555	
1E2A ;	0556	..PASS IMSBHL TO M(HIIMAG)
1E2A 4C5919;	0557	LDA ZAP1; STR MP; INC MP
1E2D 4C59;	0558	LDA ZAP1; STR MP
1E2F ;	0559	

ASP FFT Program  
Program Code

1E2F ;	0560	..-----
1E2F ;	0561	..DO THE SECOND COLUMN MULTIPLIES
1E2F ;	0562	..AND ADD TO/SUBTRACT FROM PREVIOUS DATA
1E2F ;	0563	
1E2F ;	0564	..GET R(ZAP1) TO LOCOS POINTER
1E2F F8ECAC;	0565	LDI A.0(LOCOS); PLO ZAP1
1E32 ;	0566	
1E32 ;	0567	..GET R(MQ) TO IMADD
1E32 F830BE;	0568	LDI A.1(IMADD); PHI MQ
1E35 F8F4AE;	0569	LDI A.0(IMADD); PLO MQ
1E38 ;	0570	
1E38 ;	0571	..GET R(AC) TO RLADD
1E38 ;	0572	..(DUMMY ADDRESS - HAVE TO "INP" SOMEWHERE)
1E38 F830BF;	0573	LDI A.1(RLADD); PHI AC
1E3B F8EEAF;	0574	LDI A.0(RLADD); PLO AC
1E3E ;	0575	
1E3E ;	0576	..CALL ORDER MULTIPLY
1E3E F81FB0;	0577	LDI A.1(ORDMLT); PHI ZAP
1E41 F8B5A0;	0578	LDI A.0(ORDMLT); PLO ZAP
1E44 D0;	0579	SEP ZAP ..ZAP = ORDER MULT
1E45 ;	0580	
1E45 ;	0581	..HALFWAY THROUGH GET R(FRP) TO LOREAL POINTER
1E45 F8E0A8;	0582	LDI A.0(LOREAL); PLO FRP
1E48 ;	0583	
1E48 ;	0584	..LOAD LOREAL ADDRESS INTO R(MP)
1E48 48B948A9;	0585	LDA FRP; PHI MP; LDA FRP; PLO MP
1E4C ;	0586	
1E4C ;	0587	..THEN RETURN TO GET RESULT
1E4C D0;	0588	SEP ZAP
1E4D ;	0589	
1E4D ;	0590	..ADD RESULT TO M(LOREAL)
1E4D E960;	0591	SEX MP; IRX
1E4F 8BAAF473;	0592	GLO ZAP3; PLO ZAP2; ADD; STXD
1E53 9BBA7473;	0593	GHI ZAP3; PHI ZAP2; ADC; STXD
1E57 ;	0594	
1E57 ;	0595	..-----
1E57 ;	0596	..GET R(ZAP1) TO LOSIN POINTER
1E57 F8EAAAC;	0597	LDI A.0(LOSIN); PLO ZAP1
1E5A ;	0598	
1E5A ;	0599	..GET R(MQ) TO IMADD
1E5A F8F4AE;	0600	LDI A.0(IMADD); PLO MQ
1E5D ;	0601	
1E5D ;	0602	..CALL ORDER MULTIPLY
1E5D D0;	0603	SEP ZAP
1E5E ;	0604	
1E5E ;	0605	..HALFWAY THROUGH LOAD LOIMAG ADDRESS INTO R(MP)
1E5E 48B948A9;	0606	LDA FRP; PHI MP; LDA FRP; PLO MP
1E62 ;	0607	
1E62 ;	0608	..THEN RETURN TO GET RESULT
1E62 D0;	0609	SEP ZAP
1E63 ;	0610	



```

1E63 ;                0611 ..SUBTRACT RESULT FROM M(LOIMAG)
1E63 E960;           0612         SEX MP; IRX
1E65 8BF573;         0613         GLO ZAP3; SD; STXD
1E68 9B7573;         0614         GHI ZAP3; SDB; STXD
1E6B ;              0615
1E6B ;              0616 ..-----
1E6B ;              0617 ..LOAD HIREAL ADDRESS INTO R(MP)
1E7B 48B948A9;       0618         LDA FRP; PHI MP; LDA FRP; PLO MP
1E6F ;              0619
1E6F ;              0620 ..SUBTRACT (LOCOS*IMADD) FROM M(HIREAL)
1E6F 608AF573;       0621         IRX; GLO ZAP2; SD; STXD
1E73 9A7573;         0622         GHI ZAP2; SDB; STXD
1E76 ;              0623
1E76 ;              0624 ..-----
1E76 ;              0625 ..LOAD HIIMAG ADDRESS INTO R(MP)
1E76 48B948A9;       0626         LDA FRP; PHI MP; LDA FRP; PLO MP
1E7A ;              0627
1E7A ;              0628 ..SUBTRACT (LOSIN*IMADD) FROM M(HIIMAG)
1E7A 608BF573;       0629         IRX; GLO ZAP3; SD; STXD
1E7E 9B7573;         0630         GHI ZAP3; SDB; STXD
1E81 ;              0631
1E81 ;              0632 ..-----
1E81 ;              0633 ..DO THE THIRD COLUMN MULTIPLIES
1E81 ;              0634 ..AND ADD TO/SUBTRACT FROM PREVIOUS DATA
1E81 ;              0635
1E81 ;              0636 ..GET R(ZAP1) TO LOSIN POINTER
1E81 F8EAAAC;        0637         LDI A.0(LOSIN); PLO ZAP1
1E84 ;              0638
1E84 ;              0639 ..GET R(MQ) TO RLSBLH
1E84 F8FOAE;         0640         LDI A.0(RLSBLH); PLO MQ
1E87 ;              0641
1E87 ;              0642 ..CALL ORDER MULTIPLY
1E87 DO;             0643         SEP ZAP
1E88 ;              0644
1E88 ;              0645 ..HALFWAY THROUGH GET R(FRP) TO LOREAL POINTER
1E88 F8EOA8;         0646         LDI A.0(LOREAL); PLO FRP
1E8B ;              0647
1E8B ;              0648 ..LOAD LOREAL ADDRESS INTO R(MP)
1E8B 48B948A9;       0649         LDA FRP; PHI MP; LDA FRP; PLO MP
1E8F ;              0650
1E8F ;              0651 ..THEN RETURN TO GET RESULT
1E8F DO;             0652         SEP ZAP
1E90 ;              0653
1E90 ;              0654 ..SUBTRACT RESULT FROM M(LOREAL)
1E90 E960;           0655         SEX MP; IRX
1E92 8BAAF573;       0656         GLO ZAP3; PLO ZAP2; SD; STXD
1E96 9BBA7573;       0657         GHI ZAP3; PHI ZAP2; SDB; STXD
1E9A ;              0658

```

ASP FFT Program  
Program Code

1E9A ;	0659 ..-----
1E9A ;	0660 ..GET R(ZAP1) TO LOCOS POINTER
1E9A F8ECAC;	0661 LDI A.0(LOCOS); PLO ZAP1
1E9D ;	0662
1E9D ;	0663 ..GET R(MQ) TO RLSBLH
1E9D F8FOAE;	0664 LDI A.0(RLSBLH); PLO MQ
1EAO ;	0665
1EAO ;	0666 ..CALL ORDER MULTIPLY
1EAO D0;	0667 SEP ZAP
1EA1 ;	0668
1EA1 ;	0669 ..HALFWAY THROUGH LOAD LOIMAG ADDRESS INTO R(MP)
1EA1 48B948A9;	0670 LDA FRP; PHI MP; LDA FRP; PLO MP
1EA5 ;	0671
1EA5 ;	0672 ..THEN RETURN TO GET RESULT
1EA5 D0;	0673 SEP ZAP
1EA6 ;	0674
1EA6 ;	0675 ..SUBTRACT RESULT FROM M(LOIMAG)
1EA6 E960;	0676 SEX MP; IRX
1EA8 8BF573;	0677 GLO ZAP3; SD; STXD
1EAB 9B7573;	0678 GHI ZAP3; SDB; STXD
1EAE ;	0679
1EAE ;	0680 ..-----
1EAE ;	0681 ..LOAD HIREAL ADDRESS INTO R(MP)
1EAE 48B948A9;	0682 LDA FRP; PHI MP; LDA FRP; PLO MP
1EB2 ;	0683
1EB2 ;	0684 ..ADD (LOSIN*RLSBLH) TO M(HIREAL)
1EB2 60;	0685 IRX
1EB3 8AF473;	0686 GLO ZAP2; ADD; STXD
1EB6 9A7473;	0687 GHI ZAP2; ADC; STXD
1EB9 ;	0688
1EB9 ;	0689 ..-----
1EB9 ;	0690 ..LOAD HIIMAG ADDRESS INTO R(MP)
1EB9 48B948A9;	0691 LDA FRP; PHI MP; LDA FRP; PLO MP
1EBD ;	0692
1EBD ;	0693 ..SUBTRACT (LOCOS*RLSBLH) FROM M(HIIMAG)
1EBD 60;	0694 IRX
1EBE 8BF573;	0695 GLO ZAP3; SD; STXD
1EC1 9B7573;	0696 GHI ZAP3; SDB; STXD
1EC4 ;	0697
1EC4 ;	0698



```

1EC4 ; 0699 ..*****
1EC4 ; 0700 ..INC/DEC POINTERS, CHECK FOR DONE
1EC4 ; 0701
1EC4 ; 0702 ..INCREMENT TRIG POINTERS
1EC4 ; 0703 ..GET R(FRP) TO LOCOS
1EC4 F8EDA8; 0704 LDI A.0(LOCOS+1); PLO FRP
1EC7 ; 0705
1EC7 ; 0706 ..GET R(ZAP1) TO TRIG INC
1EC7 F8E8AC; 0707 LDI A.0(TRGINC); PLO ZAP1
1ECA ; 0708
1ECA ; 0709 ..LOAD TRIG INC VALUE INTO R(ZAP2)
1ECA 4CBA4CAA; 0710 LDA ZAP1; PHI ZAP2; LDA ZAP1; PLO ZAP2
1ECE ; 0711
1ECE ; 0712 ..LOCOS <- LOCOS - TRGINC
1ECE E8; 0713 SEX FRP
1ECF 8AF573; 0714 GLO ZAP2; SD; STXD
1ED2 9A7573; 0715 GHI ZAP2; SDB; STXD
1ED5 ; 0716
1ED5 ; 0717 ..LOSIN <- LOSIN + TRGINC
1ED5 8AF4AF73; 0718 GLO ZAP2; ADD; PLO AC; STXD
1ED9 9A74BF73; 0719 GHI ZAP2; ADC; PHI AC; STXD
1EDD ; 0720
1EDD ; 0721 ..CHECK FOR LOSIN > TRGLWA (DONE CONDITION)
1EDD 8FFDFE; 0722 GLO AC; SDI A.0(TRGLWA)
1EE0 9F7D27; 0723 GHI AC; SDBI A.1(TRGLWA)
1EE3 CB1F65; 0724 LBNF FIXHLF ..IF LOSIN > TRGLWA, THEN
1EE6 ; 0725 ..GOTO FIX HALF VALUES
1EE6 ; 0726
1EE6 ; 0727
1EE6 ; 0728 ..*****
1EE6 ; 0729 ..ELSE, DECREMENT/INCREMENT FIFO POINTERS
1EE6 ; 0730
1EE6 ; 0731 ..HIIMAG <- HIIMAG - 4
1EE6 F8E7A8; 0732 LDI A.0(HIIMAG+1); PLO FRP
1EE9 E8; 0733 SEX FRP
1EEA F804F573; 0734 LDI #04; SD; STXD
1EEE F8007573; 0735 LDI #00; SDB; STXD
1EF2 ; 0736
1EF2 ; 0737 ..HIREAL <- HIREAL - 4
1EF2 F804F573; 0738 LDI #04; SD; STXD
1EF6 F8007573; 0739 LDI #00; SDB; STXD
1EFA ; 0740

```

ASP FFT Program  
Program Code

```

1EFA ;          0741 LOUP:                      ..LO---- FIFO POINTERS UP
1EFA ;          0742 ..ENTRY POINT FROM LOREAL(0)/LOIMAG(0) COMPUTE
1EFA ;          0743 ..R(FRP) @ LOIMAG.0, X @ R(FRP)
1EFA ;          0744
1EFA ;          0745 ..LOIMAG <- LOIMAG + 4
1EFA F804F473;  0746             LDI #04; ADD; STXD
1EFE F8007473;  0747             LDI #00; ADC; STXD
1FO2 ;          0748
1FO2 ;          0749 ..LOREAL <- LOREAL + 4
1FO2 F804F473;  0750             LDI #04; ADD; STXD
1FO6 F8007473;  0751             LDI #00; ADC; STXD
1FOA ;          0752
1FOA ;          0753 ..TEST FOR ADDRESSES PAST BOUNDARIES
1FOA ;          0754 ..IF IT'S A P-FFT, YOU KNOW ADDRESSES OK
1FOA ;          0755 ..CHECK P-FFT FLAG FOR TRUE(#FF)
1FOA F8C4A8;    0756             LDI A.0(PFTFLG); PLO FRP
1FOD 08CAlF62;  0757             LDN FRP; LBNZ QDCKDN ..IF TRUE, THEN
1F11 ;          0758                      ..GOTO QUAD CHECK DONE
1F11 ;          0759
1F11 ;          0760 ..ELSE IT'S A S-FFT
1F11 ;          0761 ..SET 1ST PASS FLAG FOR HI---- CHECK
1F11 F800AF;    0762             LDI #00; PLO AC
1F14 ;          0763
1F14 ;          0764 ..GET R(FRP) TO HIREAL.0
1F14 F8E5A8;    0765             LDI A.0(HIREAL+1); PLO FRP
1F17 E8;        0766             SEX FRP
1F18 ;          0767
1F18 ;          0768 HITEST:                      ..HI---- POINTER TEST
1F18 ;          0769 ..CHECK FOR HI---- < YSFWA
1F18 F801F5AE28; 0770             LDI A.0(YSFWA); SD; PLO MQ; DEC FRP
1F1D F83A75BE;  0771             LDI A.1(YSFWA); SDB; PHI MQ
1F21 C31F2D;    0772             LBDF HICHK2          ..IF HI---- => YSFWA, THEN
1F24 ;          0773                      ..GOTO CHECK FOR 2ND HI
1F24 ;          0774
1F24 ;          0775 ..ELSE, HI---- <- (NEGATIVE UNDER) + YSLWA+1
1F24 188EFC0173; 0776             INC FRP; GLO MQ; ADI A.0(YSLWA+1); STXD
1F29 9E7C3E58;  0777             GHI MQ; ADCI A.1(YSLWA+1); STR FRP
1F2D ;          0778
1F2D ;          0779 ..CHECK FOR 2ND PASS
1F2D 8FCA1F3A;  0780 HICHK2: GLO AC; LBNZ LOCHK ..IF 2ND PASS, THEN
1F31 ;          0781                      ..GOTO LO---- PTR CHECK
1F31 ;          0782
1F31 ;          0783 ..ELSE GET TO HIIMAG.0
1F31 F8E7A8;    0784             LDI A.0(HIIMAG+1); PLO FRP
1F34 ;          0785
1F34 ;          0786 ..SET 2ND PASS FLAG
1F34 F8FFAF;    0787             LDI #FF; PLO AC
1F37 ;          0788
1F37 ;          0789 ..AND GOTO RUN HIIMAG TEST
1F37 C01F18;    0790             LBR HITEST
1F3A ;          0791

```

```

1F3A ;      0792 LOCHK:                ..LO---- FIFO PTR CHECK
1F3A ;      0793 ..SET 1ST PASS FLAG FOR LO---- CHECK
1F3A F800AF; 0794          LDI #00; PLO AC
1F3D ;      0795
1F3D ;      0796 ..GET R(FRP) TO LOIMAG.0
1F3D F8E3A8; 0797          LDI A.0(LOIMAG+1); PLO FRP
1F40 ;      0798
1F40 ;      0799 LOTEST:                ..LO---- POINTER TEST
1F40 ;      0800 ..CHECK FOR LO---- => YSLWA+1
1F40 F801F5AE28; 0801          LDI A.0(YSLWA+1); SD; PLO MQ; DEC FRP
1F45 F83E75BE; 0802          LDI A.1(YSLWA+1); SDB; PHI MQ
1F49 CB1F55;  0803          LBNF LOCHK2      ..IF LO---- < YSLWA+1, THEN
1F4C ;      0804                      ..GOTO CHECK FOR 2ND LOW
1F4C ;      0805
1F4C ;      0806 ..ELSE, LO---- <- (POSITIVE OVER) + YSFWA
1F4C 188EFC0173; 0807          INC FRP; GLO MQ; ADI A.0(YSFWA); STXD
1F51 9E7C3A58; 0808          GHI MQ; ADCI A.1(YSFWA); STR FRP
1F55 ;      0809
1F55 ;      0810 ..CHECK FOR 2ND PASS
1F55 8FCA1F62; 0811 LOCHK2: GLO AC; LBNZ QDCKDN ..IF 2ND PASS, THEN
1F59 ;      0812                      ..GOTO QUAD CHECK DONE
1F59 ;      0813
1F59 ;      0814 ..ELSE GET TO LOREAL.0
1F59 F8E1A8;  0815          LDI A.0(LOREAL+1); PLO FRP
1F5C ;      0816
1F5C ;      0817 ..SET 2ND PASS FLAG
1F5C F8FFAF;  0818          LDI #FF; PLO AC
1F5F ;      0819
1F5F ;      0820 ..AND GOTO RUN LOREAL TEST
1F5F C01F40;  0821          LBR LOTEST
1F62 ;      0822
1F62 ;      0823 QDCKDN:                ..QUAD CHECK DONE
1F62 ;      0824 ..ADDRESSES ALL OK BY BOUNDARIES
1F62 ;      0825 ..GOTO COMPUTE THESE MIDDLE SIX VALUES
1F62 C01DE6;  0826          LBR MIDSIX
1F65 ;      0827
1F65 ;      0828

```

ASP FFT Program  
Program Code

```

1F65 ;      0829 ..*****
1F65 ;      0830 FIXHLF:                ..FIX HALF VALUES
1F65 ;      0831 ..YOU HAVEN'T COMPUTED THE VALUES CORRECTLY
1F65 ;      0832 ..      FOR THE M INDICIES
1F65 ;      0833 ..IT HAPPENS THAT THE REAL(M) VALUE IS OK (LUCK)
1F65 ;      0834 ..BUT IMAG(M) IS TWICE WHAT IT SHOULD BE
1F65 ;      0835 ..DIVIDE IT BY 2 FOR THE CORRECT VALUE
1F65 ;      0836 ..(WITH SIGN EXTEND)
1F65 ;      0837
1F65 ;      0838 ..M(LOIMAG) STILL HAS THE CORRECT ADDRESS
1F65 F8E2A8; 0839      LDI A.0(LOIMAG); PLO FRP
1F68 48B908A9; 0840      LDA FRP; PHI MP; LDN FRP; PLO MP
1F6C 09FE;    0841      LDN MP; SHL
1F6E CB1F73;  0842      LBNF HLFNEG      ..IF NEG, THEN
1F71 F901;    0843      ORI #01      ..SET SIGN EXTEND
1F73 ;        0844 HLFNEG:      ..HALF VALUE NEGATIVE
1F73 7676;    0845      RSHR; RSHR
1F75 5919;    0846      STR MP; INC MP
1F77 097659;  0847      LDN MP; SHRC; STR MP
1F7A ;        0848
1F7A ;        0849
1F7A ;        0850 ..*****
1F7A ;        0851 NDORDR:      ..END ORDER
1F7A D5;      0852      EXIT
1F7B ;        0853
1F7B C4C4C4C4C4C4; 0854      ,#C4C4C4C4C4C4
1F81 ;        0855

```



```

1F81 ;      0856 ..#####
1F81 ;      0857 ..COMBO.SR                ..COMBINE LO/HI-REAL/IMAG
1F81 ;      0858 ..
1F81 ;      0859 ..THIS ROUTINE COMPUTES HI - LO VALUE
1F81 ;      0860 ..                LO - HI VALUE
1F81 ;      0861 ..                LO + HI VALUE
1F81 ;      0862 ..AND STORES THEM
1F81 ;      0863 ..
1F81 ;      0864 ..IT ASSUMES R(FRP) IS AT LO PTR, R(ZAP1) IS AT
1F81 ;      0865 ..HI PTR, AND R(ZAP3) IS AT HI-LO ADDRESS (+1)
1F81 ;      0866 ..
1F81 ;      0867 ..ON RETURN R(FRP) & R(ZAP1) ARE INCREMENTED BY 2
1F81 ;      0868 ..AND R(ZAP3) IS AT LO-HI + 1
1F81 ;      0869 ..#####
1F81 ;      0870
1F81 ;      0871 ..RETURN PORTION
1F81 ;      0872 EXCMBO:                ..EXIT COMBO
1F81 D3;      0873         SEP PC                ..RESET TO R(PC)
1F82 ;      0874
1F82 ;      0875 ..ENTER HERE
1F82 ;      0876 COMBO:                ..COMBINE VALUES
1F82 ;      0877 ..GET R(MP) TO LO---- ADDRESS
1F82 48B948A9; 0878         LDA FRP; PHI MP; LDA FRP; PLO MP
1F86 ;      0879
1F86 ;      0880 ..GET R(ZAP2) TO HI---- ADDRESS
1F86 4CBA4CAA; 0881         LDA ZAP1; PHI ZAP2; LDA ZAP1; PLO ZAP2
1F8A ;      0882
1F8A ;      0883 ..GET LO---- VALUE INTO R(AC)
1F8A 49BF49AF; 0884         LDA MP; PHI AC; LDA MP; PLO AC
1F8E ;      0885
1F8E ;      0886 ..DO HI-LO AND STORE
1F8E EA;      0887         SEX ZAP2
1F8F 1A8FF5;  0888         INC ZAP2; GLO AC; SD
1F92 5B2B;    0889         STR ZAP3; DEC ZAP3
1F94 2A9F75;  0890         DEC ZAP2; GHI AC; SDB
1F97 5B2B;    0891         STR ZAP3; DEC ZAP3
1F99 ;      0892
1F99 ;      0893 ..DO LO-HI AND STORE
1F99 1A8FF7;  0894         INC ZAP2; GLO AC; SM
1F9C 5B2B;    0895         STR ZAP3; DEC ZAP3
1F9E 2A9F77;  0896         DEC ZAP2; GHI AC; SMB
1FA1 5B2B;    0897         STR ZAP3; DEC ZAP3
1FA3 ;      0898
1FA3 ;      0899 ..DO LO+HI AND STORE
1FA3 1A8FF4;  0900         INC ZAP2; GLO AC; ADD
1FA6 5B2B;    0901         STR ZAP3; DEC ZAP3
1FA8 2A9F74;  0902         DEC ZAP2; GHI AC; ADC
1FAB 5B2B;    0903         STR ZAP3; DEC ZAP3
1FAD ;      0904
1FAD ;      0905 ..THEN RETURN THRU TOP
1FAD C01F81;  0906         LBR EXCMBO

```



ASP FFT Program  
Program Code

```

LFBO ;          0907
LFBO ;          0908 ..#####
LFBO ;          0909 ..ORDMLT.SR      M(M(R(ZAP1))) * M(R(MQ)) -> R(ZAP3)
LFBO ;          0910 ..
LFBO ;          0911 ..THIS PROGRAM ASSUMES ZAP1 POINTS TO OPERAND A PTR
LFBO ;          0912 ..                                MQ POINTS TO OPERAND B
LFBO ;          0913 ..AND AC POINTS TO A DUMMY LOCATION
LFBO ;          0914 ..( FOR INP OPERATION)
LFBO ;          0915 ..
LFBO ;          0916 ..IT DELIVERS THE OPERANDS TO THE MULTIPLIER
LFBO ;          0917 ..JUMPS OUT WHILE THE HARDWARE OPERATES
LFBO ;          0918 ..THEN RETURNS TO STORE THE RESULT
LFBO ;          0919 ..SINCE ONE OPERAND IS A FRACTION
LFBO ;          0920 ..THE RESULT IS SCALED BY 2 TO BE CORRECT
LFBO ;          0921 ..
LFBO ;          0922 ..MA, MQ ARE LEFT INCREMENTED BY 2
LFBO ;          0923 ..
LFBO ;          0924 ..NOTE: P MUST BE ZAP(RO)
LFBO ;          0925 ..#####
LFBO ;          0926
LFBO ;          0927
LFBO ;          0928 EXITM:                ..EXIT ORDER MULTIPLY
LFBO EO;          0929                SEX ZAP                ..OUT IMMEDIATE
LFB1 6100;        0930                OUT 1,#00            ..TURN OFF MULTIPLIER
LFB3 ;           0931
LFB3 ;           0932 ..ENABLE INTERRUPTS AND RETURN
LFB3 7003;        0933                RET ,#03            ..IE=1, X=0, P=3
LFB5 ;           0934
LFB5 ;           0935 ORDMLT:                ..ORDER MULTIPLY ROUTINE
LFB5 ;           0936 ..GET R(MA) TO OPA ADDRESS
LFB5 4CBD4CAD;    0937                LDA ZAP1; PHI MA; LDA ZAP1; PLO MA
LFB9 ;           0938
LFB9 ;           0939 ..DISABLE INTERRUPTS
LFB9 EO;          0940                SEX ZAP                ..OUT IMMEDIATE
LFBA 7100;        0941                DIS ,#00            ..IE=0, X=0, P=0
LFBC ;           0942
LFBC ;           0943 ..OUTPUT OPERANDS
LFBC 6110;        0944                OUT 1,#10            ..LATCH SEL4
LFBE 6200;        0945                OUT 2,#00            ..RESET MULTIPLIER
LFC0 ED;          0946                SEX MA                ..SET X TO MA FOR OPA OUT
LFC1 64;          0947                OUT 4                ..OPA.1 TO MULTIPLIER
LFC2 63;          0948                OUT 3                ..OPA.0 TO MULTIPLIER
LFC3 EE;          0949                SEX MQ                ..SET X TO MQ FOR OPB OUT
LFC4 66;          0950                OUT 6                ..OPB.1 TO MULTIPLIER
LFC5 65;          0951                OUT 5                ..OPB.0 TO MULTIPLIER
LFC6 EO;          0952                SEX ZAP                ..OUT IMMEDIATE
LFC7 6700;        0953                OUT 7,#00            ..EXECUTE
LFC9 ;           0954
LFC9 ;           0955 ..NOW RETURN TO MAIN PROGRAM WHILE HARDWARE WORKS
LFC9 D3;          0956                SEP PC
LFCA ;           0957

```

ASP FFT Program  
Program Code

```

1FCA ;          0958 ..WHEN YOU COME BACK, STORE THE RESULT
1FCA E1F;       0959     SEX AC; INC AC    ..SET X TO AC FOR RESULT IN
1FCC 6BFE;      0960     INP 3; SHL      ..GET LOW * 2
1FCE AB73;      0961     PLO ZAP3; STXD   ..SAVE & STORE
1FD0 6C7E;      0962     INP 4; SHLC    ..GET HIGH * 2
1FD2 BB5F;      0963     PHI ZAP3; STR AC ..SAVE & STORE
1FD4 ;         0964
1FD4 ;         0965 ..AND RETURN THROUGH TOP
1FD4 30B0;      0966     BR EXITM
1FD6 ;         0967
1FD6 ;         0968 ..#####
1FD6 ;         0969 ..END OF ORDER.SR

```

ASP FFT Program  
Program Code

A.2.2 FIT Code

A.2.2.1 MAGAPX (Magnitude Approximate)

```

0000 ;          0001
0000 ;          0002 ..MAGAPX.SR          20 JAN 80          9:20 PM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..MAGAPX.SR          ..MAGNITUDE APPROXIMATE
0000 ;          0244 ..
0000 ;          0245 ..THIS ROUTINE APPROXIMATES THE VALUE FOR MAGNITUDE
0000 ;          0246 ..BY THE ALGORITHM:
0000 ;          0247 ..      IF G(N)/4 => L(N)
0000 ;          0248 ..      THEN M(N) <- G(N)
0000 ;          0249 ..      ELSE M(N) <- 0.7659 * [G(N) + L(N)]
0000 ;          0250 ..WHERE:
0000 ;          0251 ..      G(N) <- GREATEST [ ABS.R(N) OR ABS.I(N) ]
0000 ;          0252 ..      L(N) <- LEAST   [ ABS.R(N) OR ABS.I(N) ]
0000 ;          0253 ..#####
0000 ;          0254
0000 ;          0255
0000 ;          0256 ..EXTERNAL ROUTINES
0000 ;          0257
0000 ;          0258          SUBROT=#2800          ..SUBROUTINE BLOCK #1
0000 ;          0259          SNMULT=SUBROT+4        ..SINE MULTIPLY
0000 ;          0260
0000 ;          0261          SUBRT2=#2B50          ..SUBROUTINE BLOCK #2
0000 ;          0262          BDRYST=SUBRT2+4          ..BOUNDARY SET
0000 ;          0263          BDRYCK=SUBRT2+6          ..BOUNDARY CHECK
0000 ;          0264
0000 ;          0265          DTASCL=#2960          ..DATA SCALE
0000 ;          0266

```

```

0000 ;          0267 ..BEGIN HERE
0000 ;          0268
0000 ;          0269          ORG #2000
2000 ;          0270
2000 ;          0271 MAGAPX:          ..MAGNITUDE APPROXIMATE
2000 ;          0272
2000 ;          0273 ..*****
2000 ;          0274 ..SCALE VALUES TO MAX #3FFF (OR #C000)
2000 ;          0275
2000 ;          0276 ..GET R(MP) TO BASE ADDRESS
2000 F830B9;    0277          LDI A.1(BASE); PHI MP
2003 F8D8A9;    0278          LDI A.0(BASE); PLO MP
2006 ;          0279
2006 ;          0280 ..GET R(FRP) TO FFT-LENGTH
2006 F8D0A8;    0281          LDI A.0(FFTLN); PLO FRP
2009 ;          0282
2009 ;          0283 ..GET R(ZAP1) TO YSLWA+1
2009 F8B0AC;    0284          LDI A.0(YSTBL+2); PLO ZAP1
200C ;          0285
200C ;          0286 ..LOAD MAXIMUM SHIFT VALUE INTO R(ZAP2)
200C ;          0287 ..THAT IS: 6 (TO LEAVE HI-BIT FREE)
200C F800BA;    0288          LDI #00; PHI ZAP2
200F F806AA;    0289          LDI #06; PLO ZAP2
2012 ;          0290
2012 ;          0291 ..CALL DATA SCALE TO SCALE FIFO
2012 F829B0;    0292          LDI A.1(DTASCL); PHI ZAP
2015 F860A0;    0293          LDI A.0(DTASCL); PLO ZAP
2018 D0;        0294          SEP ZAP          ..ZAP = DATA SCALE
2019 C4C4C4;    0295          ,#C4C4C4
201C ;          0296
201C ;          0297 ..*****
201C ;          0298 ..INITILIZE FOR BOUNDARY SET
201C ;          0299 ..GET INPUT POINTER, R(MA), TO BASE ADDRESS
201C F8D8AC;    0300          LDI A.0(BASE); PLO ZAP1
201F 4CBD;      0301          LDA ZAP1; PHI MA
2021 4CAD;      0302          LDA ZAP1; PLO MA
2023 ;          0303
2023 ;          0304 ..GET R(ZAP1) TO YSLWA+1 POINTER
2023 F8B0AC;    0305          LDI A.0(YSTBL+2); PLO ZAP1
2026 ;          0306
2026 ;          0307 ..GET R(MP) TO FFT LENGTH
2026 F830B9;    0308          LDI A.1(FFTLN); PHI MP
2029 F8D0A9;    0309          LDI A.0(FFTLN); PLO MP
202C ;          0310
202C ;          0311 ..CALL BOUNDARY SET FOR BDRYOK & FREELN
202C F82BB0;    0312          LDI A.1(BDRYST); PHI ZAP
202F F854A0;    0313          LDI A.0(BDRYST); PLO ZAP
2032 D0;        0314          SEP ZAP          ..ZAP = BOUNDARY SET
2033 ;          0315

```



ASP FFT Program  
Program Code

2033 ;	0316 ..*****
2033 ;	0317 ..SET UP FOR MAGNITUDE APPROXIMATE
2033 ;	0318 ..GET R(MQ) TO FACTOR
2033 F820BE;	0319 LDI A.1(FACTOR); PHI MQ
2036 F8DBAE;	0320 LDI A.0(FACTOR); PLO MQ
2039 ;	0321
2039 ;	0322 ..-----
2039 ;	0323 NEXTMN: ..NEXT M(N) APPROXIMATE
2039 ;	0324 ..GET R(AC) TO R(N) ADDRESS
2039 9DBF8DAF;	0325 GHI MA; PHI AC; GLO MA; PLO AC
203D ;	0326
203D ;	0327 ..CHECK FOR R(N) POSITIVE
203D 4D;	0328 LDA MA
203E FECB204E;	0329 SHL; LBNF UPTOIN ..IF POSITIVE, THEN
2042 ;	0330 ..GO UP TO I(N)
2042 ;	0331
2042 ;	0332 ..ELSE INVERT R(N) AND STORE BACK
2042 76AA;	0333 SHRC; PLO ZAP2
2044 ODFD005D2D;	0334 LDN MA; SDI #00; STR MA; DEC MA
2049 8A7D005D1D;	0335 GLO ZAP2; SDBI #00; STR MA; INC MA
204E ;	0336
204E ;	0337 UPTOIN: ..UP TO I(N)
204E ;	0338 ..GET TO I(N)
204E 1D;	0339 INC MA
204F ;	0340
204F ;	0341 ..CALL BOUNDARY CHECK TO CHECK I(N) FOR YPLWA+1
204F F856A0;	0342 LDI A.0(BDRYCK); PLO ZAP
2052 D0;	0343 SEP ZAP ..ZAP = BOUNDARY CHECK
2053 ;	0344
2053 ;	0345 ..LOAD ABS.I(N) INTO R(ZAP2)
2053 4D;	0346 LDA MA
2054 FEC3205F;	0347 SHL; LBDF ININV ..IF NEGATIVE, THEN
2058 ;	0348 ..GOTO I(N) INVERT
2058 ;	0349
2058 ;	0350 ..ELSE LOAD STRAIGHT
2058 76BA;	0351 SHRC; PHI ZAP2
205A ODAA;	0352 LDN MA; PLO ZAP2
205C ;	0353 ..AND LEAVE R(MA) AT OLD I(N).0
205C ;	0354
205C ;	0355 ..THEN GO TO COMPARE R(N) & I(N)
205C C02069;	0356 LBR CMPRRI
205F ;	0357
205F ;	0358 ININV: ..I(N) INVERT
205F ;	0359 ..PUT INVERTED I(N) INTO R(ZAP2)
205F 76BA;	0360 SHRC; PHI ZAP2
2061 ODFD00AA;	0361 LDN MA; SDI #00; PLO ZAP2
2065 9A7D00BA;	0362 GHI ZAP2; SDBI #00; PHI ZAP2
2069 ;	0363 ..AGAIN LEAVE R(MA) AT OLD I(N).0
2069 ;	0364 ..STILL FALL THROUGH TO COMPARE R(N) & I(N)
2069 ;	0365



2069 ;	0366 .....
2069 ;	0367 CMPRRI: ..COMPARE R(N) & I(N)
2069 ;	0368 ..CHECK FOR ABS.R(N) => ABS.I(N)
2069 EFLF;	0369 SEX AC; INC AC
206B 8AF52F;	0370 GLO ZAP2; SD; DEC AC
206E 9A75;	0371 GHI ZAP2; SDB
2070 CB208F;	0372 LBNF IGTR ..IF NOT TRUE, THEN
2073 ;	0373 ..GOTO I(N) > R(N)
2073 ;	0374
2073 ;	0375 ..IF TRUE GET [ABS.R(N)]/4 INTO OLD I(N)
2073 4FF6BB;	0376 LDA AC; SHR; PHI ZAP3
2076 OF76AB2F;	0377 LDN AC; SHRC; PLO ZAP3; DEC AC
207A 9BF62D5D;	0378 GHI ZAP3; SHR; DEC MA; STR MA
207E 8B761D5D;	0379 GLO ZAP3; SHRC; INC MA; STR MA
2082 ;	0380
2082 ;	0381 ..CHECK [ABS.R(N)]/4 => ABS.I(N)
2082 ED;	0382 SEX MA
2083 8AF52D;	0383 GLO ZAP2; SD; DEC MA
2086 9A751D;	0384 GHI ZAP2; SDB; INC MA
2089 CB20AD;	0385 LBNF FCTXSM ..IF NOT TRUE, THEN
208C ;	0386 ..GOTO FACTOR TIMES SUM
208C ;	0387
208C ;	0388 ..IF TRUE GO TO END/DONE CHECK (G(N) IS R(N), OK)
208C C020C3;	0389 LBR NDDNCK
208F ;	0390
208F ;	0391 IGTR: ..ABS.I(N) > ABS.R(N)
208F ;	0392 ..GET [ABS.I(N)]/4 INTO R(ZAP3)
208F 9AF6BB;	0393 GHI ZAP2; SHR; PHI ZAP3
2092 8A76AB;	0394 GLO ZAP2; SHRC; PLO ZAP3
2095 9BF6BB;	0395 GHI ZAP3; SHR; PHI ZAP3
2098 8B76AB;	0396 GLO ZAP3; SHRC; PLO ZAP3
209B ;	0397
209B ;	0398 ..CHECK [ABS.I(N)]/4 => ABS.R(N)
209B EFLF;	0399 SEX AC; INC AC
209D 8BF72F;	0400 GLO ZAP3; SM; DEC AC
20A0 9B77;	0401 GHI ZAP3; SMB
20A2 CB20AD;	0402 LBNF FCTXSM ..IF NOT TRUE, THEN
20A5 ;	0403 ..GOTO FACTOR TIMES SUM
20A5 ;	0404
20A5 ;	0405 ..IF TRUE STORE ABS.I(N) INTO M(R(AC))
20A5 9A5F1F;	0406 GHI ZAP2; STR AC; INC AC
20A8 8A5F;	0407 GLO ZAP2; STR AC
20AA C020C3;	0408 LBR NDDNCK ..THEN GOTO END/DONE CHECK

ASP FFT Program  
Program Code

20AD ;	0409 ..-----
20AD ;	0410 FCTXSM: ..FACTOR TIMES SUM
20AD ;	0411 ..GET SUM INTO OLD I(N).0
20AD EF1F;	0412 SEX AC; INC AC
20AF 8AF45D;	0413 GLO ZAP2; ADD; STR MA
20B2 2F2D;	0414 DEC AC; DEC MA
20B4 9A745D;	0415 GHI ZAP2; ADC; STR MA
20B7 ;	0416
20B7 ;	0417 ..THEN CALL SINE MULTIPLY
20B7 ;	0418 ..WITH SUM @ M(R(MA)) & FACTOR @ M(R(MQ))
20B7 F828B0;	0419 LDI A.1(SNMULT); PHI ZAP
20BA F804A0;	0420 LDI A.0(SNMULT); PLO ZAP
20BD D0;	0421 SEP ZAP ..ZAP = SINE MULTIPLY
20BE ;	0422
20BE ;	0423 ..AT HALFWAY, RETURN R(MA) TO OLD I(N).0
20BE 2D;	0424 DEC MA
20BF ;	0425
20BF ;	0426 ..RETURN R(MQ) TO FACTOR
20BF 2E2E;	0427 DEC MQ; DEC MQ
20C1 C4;	0428 NOP ..AND WASTE TIME
20C2 ;	0429
20C2 ;	0430 ..THEN RETURN TO STORE RESULT @ M(R(AC))
20C2 D0;	0431 SEP ZAP ..ZAP = MIDDLE OF SINE MULT
20C3 ;	0432
20C3 ;	0433 ..THEN FALL THROUGH TO END/DONE CHECK
20C3 ;	0434
20C3 ;	0435 ..-----
20C3 ;	0436 NDDNCK: ..END/DONE CHECK
20C3 ;	0437 ..ZERO OLD I(N), FOR FUTURE SCALINGS
20C3 2DF800;	0438 DEC MA; LDI #00
20C6 5D1D;	0439 STR MA; INC MA
20C8 5D1D;	0440 STR MA; INC MA
20CA ;	0441 ..AND LEAVE R(MA) AT NEXT (SUPPOSED) R(N)
20CA ;	0442
20CA ;	0443 ..THEN CALL BOUNDARY CHECK
20CA ;	0444 ..TO TEST FOR YSLWA+1 AND DONE
20CA F82BB0;	0445 LDI A.1(BDRYCK); PHI ZAP
20CD F856A0;	0446 LDI A.0(BDRYCK); PLO ZAP
20D0 D0;	0447 SEP ZAP ..ZAP = BOUNDARY CHECK
20D1 ;	0448
20D1 ;	0449 ..ON RETURN CHECK D FOR DONE FLAG
20D1 CA2039;	0450 LBNZ NEXTMN ..IF NOT DONE, THEN
20D4 ;	0451 ..GOTO NEXT M(N)
20D4 ;	0452
20D4 ;	0453 ..ELSE DONE WITH MAG APPROXIMATE, RETURN TO MAIN
20D4 D5;	0454 EXIT
20D5 C4C4C4C4C4C4;	0455 ,#C4C4C4C4C4C4
20DB ;	0456

ASP FFT Program  
Program Code

```

20DB ; 0457 ..*****
20DB ; 0458 FACTOR: ..FACTOR
20DB 6209; 0459 ,#6209 ..DECIMAL 0.7659
20DD ; 0460 ..AVG [ 1/(SIN + COS) ] FOR
20DD ; 0461 ..ARCTAN(1/4) & ARCTAN(1/1)
20DD ; 0462
20DD ; 0463 ..#####
20DD ; 0464 ..END OF MAGAPX.SR

```

ASP FFT Program  
Program Code

A.2.2.2 SMTGMX (Smooth and Tag Maximum)

```

0000 ;          0001
0000 ;          0002 ..SMTGMX.SR          25 MAY 80          10:55 AM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..SMTGMX.SR          SMOOTH & TAG MAXIMUM
0000 ;          0244 ..
0000 ;          0245 ..THIS ROUTINE DOES A RUNNING 8-POINT AVERAGE
0000 ;          0246 ..TO SMOOTH THE FREQUENCY COMPONENTS
0000 ;          0247 ..AND TAGS THE GREATEST VALUE (AFTER SMOOTHING)
0000 ;          0248 ..
0000 ;          0249 ..IT ASSUMES THAT THE MAGNITUDE VALUES ARE IN EVERY
0000 ;          0250 ..OTHER LOCATION IN THE FIFO STARTING WITH THE BASE
0000 ;          0251 ..ADDRESS (AND #0000 IN BETWEEN)
0000 ;          0252 ..
0000 ;          0253 ..IT RETURNS WITH THE SMOOTHED VALUES IN THOSE
0000 ;          0254 ..LOCATIONS, AND:
0000 ;          0255 ..      THE ADDRESS OF THE MAXIMUM IN MXADDR
0000 ;          0256 ..      THE FREQUENCY INDEX OF THE MAXIMUM IN FMAX
0000 ;          0257 ..      & THE VALUE OF THE MAXIMUM IN MXVALU
0000 ;          0258 ..#####
0000 ;          0259
0000 ;          0260
0000 ;          0261 ..EXTERNAL ROUTINES
0000 ;          0262
0000 ;          0263          SUBRT2=#2B50          ..SUBROUTINE BLOCK #2
0000 ;          0264          BDRYST=SUBRT2+4          ..BOUNDARY SET
0000 ;          0265          DTASCL=#2960          ..DATA SCALE
0000 ;          0266
0000 ;          0267 ..INTERNAL VARIABLES (FROM NOW ON)
0000 ;          0268
0000 ;          0269          MXADDR=CSVLU+2          ..ADDRESS OF MAXIMUM
0000 ;          0270          FMAX=MXADDR+2          ..FREQUENCY INDEX OF MAX
0000 ;          0271          MXVALU=FMAX+2          ..MAXIMUM VALUE (INTEGER)
0000 ;          0272          MXEXP=MXVALU+2          ..EXPONENT (SHIFT) VALUE
0000 ;          0273          ..OF MAX
0000 ;          0274

```



```

0000 ;      0275 ..BEGIN HERE
0000 ;      0276
0000 ;      0277          ORG #2100
2100 ;      0278
2100 ;      0279 SMTGMX:          ..SMOOTH & TAG MAXIMUM
2100 ;      0280
2100 ;      0281 ..*****
2100 ;      0282 ..SCALE VALUES TO MAX #7FFF
2100 ;      0283 ..GET R(MP) TO BASE ADDRESS
2100 F830B9; 0284          LDI A.1(BASE); PHI MP
2103 F8D8A9; 0285          LDI A.0(BASE); PLO MP
2106 ;      0286
2106 ;      0287 ..GET R(FRP) TO FFT-LENGTH
2106 F8D0A8; 0288          LDI A.0(FFTLN); PLO FRP
2109 ;      0289
2109 ;      0290 ..GET R(ZAP1) TO YSLWA+1
2109 F8B0AC; 0291          LDI A.0(YSTBL+2); PLO ZAP1
210C ;      0292
210C ;      0293 ..LOAD MAXIMUM SHIFT VALUE INTO R(ZAP2)
210C ;      0294 ..THAT IS: 7 (SHIFT TO MAXIMUM)
210C F800BA; 0295          LDI #00; PHI ZAP2
210F F807AA; 0296          LDI #07; PLO ZAP2
2112 ;      0297
2112 ;      0298 ..CALL DATA SCALE TO SCALE FIFO
2112 F829B0; 0299          LDI A.1(DTASCL); PHI ZAP
2115 F860A0; 0300          LDI A.0(DTASCL); PLO ZAP
2118 D0;     0301          SEP ZAP          ..ZAP = DATA SCALE
2119 C4C4C4; 0302          ,#C4C4C4
211C ;      0303
211C ;      0304 ..*****
211C ;      0305 ..DO BOUNDARY SET FOR BOUNDARY OK FLAG & FREE LENGTH
211C ;      0306 ..GET INPUT POINTER, R(MA), TO BASE ADDRESS
211C F8D8AC; 0307          LDI A.0(BASE); PLO ZAP1
211F 4CBD;   0308          LDA ZAP1; PHI MA
2121 4CAD;   0309          LDA ZAP1; PLO MA
2123 ;      0310
2123 ;      0311 ..GET R(ZAP1) TO YSLWA+1 POINTER
2123 F8B0AC; 0312          LDI A.0(YSTBL+2); PLO ZAP1
2126 ;      0313
2126 ;      0314 ..GET R(MP) TO FFT LENGTH
2126 F830B9; 0315          LDI A.1(FFTLN); PHI MP
2129 F8D0A9; 0316          LDI A.0(FFTLN); PLO MP
212C ;      0317
212C ;      0318 ..CALL BOUNDARY SET FOR BDRYOK & FREELN
212C F82BB0; 0319          LDI A.1(BDRYST); PHI ZAP
212F F854A0; 0320          LDI A.0(BDRYST); PLO ZAP
2132 D0;     0321          SEP ZAP          ..ZAP = BOUNDARY SET
2133 ;      0322

```

ASP FFT Program  
Program Code

2133 ;	0323 ..*****
2133 ;	0324 ..SET SUBROUTINE POINTER TO SMOOTH BOUNDARY CHECK
2133 F822B0;	0325 LDI A.1(SMBDCK); PHI ZAP
2136 F806A0;	0326 LDI A.0(SMBDCK); PLO ZAP
2139 ;	0327
2139 ;	0328 ..SET OUTPUT POINTER R(MP) AND MAX POINTER R(ZAP3)
2139 ;	0329 ..TO INPUT POINTER R(MA)
2139 9DB9BB;	0330 GHI MA; PHI MP; PHI ZAP3
213C 8DA9AB;	0331 GLO MA; PLO MP; PLO ZAP3
213F ;	0332
213F ;	0333 ..ADD 1ST EIGHT VALUES
213F ;	0334 ..SET HEAD COUNTER TO 8
213F F808A8;	0335 LDI #08; PLO FRP
2142 ;	0336
2142 ;	0337 ..SET R(MQ.O,AC) 24-BIT ACCUMULATOR TO #000000
2142 F800AEBFAF;	0338 LDI #00; PLO MQ; PHI AC; PLO AC
2147 ;	0339
2147 ;	0340 ADDNXT: ..ADD NEXT FREQUENCY VALUE
2147 ;	0341 ..ADD FROM INPUT POINTER TO ACCUMULATOR
2147 ED1D;	0342 SEX MA; INC MA
2149 8FF4AF2D;	0343 GLO AC; ADD; PLO AC; DEC MA
214D 9F74BF;	0344 GHI AC; ADC; PHI AC
2150 CB21541E;	0345 LBNF *+4; INC MQ ..IF OVERFLOW, THEN
2154 ;	0346 ..INCREMENT HIGH BYTE
2154 ;	0347
2154 ;	0348 ..GET TO NEXT FREQUENCY MAGNITUDE
2154 1D1D1D1D;	0349 INC MA; INC MA; INC MA; INC MA
2158 ;	0350
2158 ;	0351 ..DECREMENT HEAD COUNT, CHECK FOR DONE
2158 2888;	0352 DEC FRP; GLO FRP
215A C22161;	0353 LBZ SAVAVG ..IF DONE, THEN
215D ;	0354 ..GOTO SAVE AVERAGE
215D ;	0355
215D ;	0356 ..ELSE CHECK INPUT POINTER FOR YSLWA+1 OR +3
215D D0;	0357 SEP ZAP
215E ;	0358
215E ;	0359 ..THEN GO BACK FOR NEXT ADD
215E C02147;	0360 LBR ADDNXT
2161 ;	0361

```

2161 ; 0362 ..-----
2161 ; 0363 ..MAIN BODY OF SMOOTH
2161 ; 0364 ..SAVE AVG, BOUNDARY CHECK, SUBTRACT FROM OUTPUT,
2161 ; 0365 ..STORE SUM, TEST FOR MAX & ADD FROM INPUT
2161 ; 0366
2161 ; 0367 SAVAVG: ..SAVE AVERAGE
2161 ; 0368 ..AVERAGE SUM OF 8, AND SAVE (EVENTUALLY) IN R(ZAP2)
2161 8EF6AC; 0369 GLO MQ; SHR; PLO ZAP1
2164 9F76BA; 0370 GHI AC; SHRC; PHI ZAP2
2167 8F76AA; 0371 GLO AC; SHRC; PLO ZAP2 ..DIVIDE ONCE
216A ; 0372
216A 8CF6AC; 0373 GLO ZAP1; SHR; PLO ZAP1
216D 9A76BA; 0374 GHI ZAP2; SHRC; PHI ZAP2
2170 8A76AA; 0375 GLO ZAP2; SHRC; PLO ZAP2 ..DIVIDE TWICE
2173 ; 0376
2173 9AF6BA; 0377 GHI ZAP2; SHR; PHI ZAP2
2176 8A76AA; 0378 GLO ZAP2; SHRC; PLO ZAP2 ..DIVIDE THRICE
2179 ; 0379
2179 ; 0380 ..SUBTRACT OLD VALUE FROM 24-BIT ACCUMULATOR
2179 E919; 0381 SEX MP; INC MP
217B 8FF7AF29; 0382 GLO AC; SM; PLO AC; DEC MP
217F 9F77BF; 0383 GHI AC; SMB; PHI AC
2182 C321862E; 0384 LBDF *+4; DEC MQ ..IF BORROWED, THEN
2186 ; 0385 ..DECREMENT HIGH BYTE
2186 ; 0386
2186 ; 0387 ..STORE SAVED AVERAGE THERE
2186 9A5919; 0388 GHI ZAP2; STR MP; INC MP
2189 8A5919; 0389 GLO ZAP2; STR MP; INC MP
218C ; 0390
218C ; 0391 ..TEST TO ASSURE
218C ; 0392 ..OLD MAX @ M(R(ZAP3)) > NEW SMOOTHED @ R(ZAP2)
218C EB1B; 0393 SEX ZAP3; INC ZAP3
218E 8AF72B; 0394 GLO ZAP2; SM; DEC ZAP3
2191 9A77; 0395 GHI ZAP2; SMB
2193 CB219E; 0396 LBNF GOZOUT ..IF OLD > NEW, THEN
2196 ; 0397 ..GOTO GOOSE OUTPUT UP BY 4
2196 ; 0398
2196 ; 0399 ..ELSE NEW => OLD, SAVE NEW ADDRESS AS MAX
2196 89FF02AB; 0400 GLO MP; SMI #02; PLO ZAP3
219A 997F00BB; 0401 GHI MP; SMI #00; PHI ZAP3
219E ; 0402
219E ; 0403 GOZOUT: ..GOOSE OUTPUT UP BY 2
219E ; 0404 ..GET OUTPUT POINTER TO NEXT MAGNITUDE LOCATION
219E 1919; 0405 INC MP; INC MP
21A0 ; 0406
21A0 ; 0407 ..CHECK INPUT AND OUTPUT POINTER FOR YSLWA+1 & +3
21A0 D0; 0408 SEP ZAP
21A1 ; 0409
21A1 ; 0410 ..ON RETURN CHECK D FOR DONE FLAG
21A1 C221B8; 0411 LBZ SMMXDN ..IF DONE, THEN
21A4 ; 0412 ..GOTO SMOOTH/MAX DONE

```



ASP FFT Program  
Program Code

21A4 ;	0413
21A4 ;	0414 ..NOW ADD NEW VALUE FROM INPUT POINTER
21A4 ED1D;	0415 SEX MA; INC MA
21A6 8FF4AF2D;	0416 GLO AC; ADD; PLO AC; DEC MA
21AA 9F74BF;	0417 GHI AC; ADC; PHI AC
21AD CB21B11E;	0418 LBNF *+4; INC MQ ..IF OVERFLOW, THEN
21B1 ;	0419 ..INCREMENT HIGH BYTE
21B1 ;	0420
21B1 ;	0421 ..GET TO NEXT FREQUENCY MAGNITUDE
21B1 1D1D1D1D;	0422 INC MA; INC MA; INC MA; INC MA
21B5 ;	0423
21B5 ;	0424 ..THEN GOTO SAVE AVERAGE OF LAST 8
21B5 C02161;	0425 LBR SAVAVG
21B8 ;	0426
21B8 ;	0427 ..-----
21B8 ;	0428 SMMXDN: ..SMOOTH/MAXIMUM DONE
21B8 ;	0429 ..ALL VALUES SMOOTHED AND IN FIFO
21B8 ;	0430 ..(EVERY 4TH LOCATION, FOR 2 BYTES)
21B8 ;	0431 ..MAXIMUM VALUE AT M(R(ZAP3))
21B8 ;	0432
21B8 ;	0433 ..STORE ADDRESS OF MAXIMUM INTO MEMORY
21B8 F8EOA8;	0434 LDI A.0(MXADDR); PLO FRP
21BB 9B5818;	0435 GHI ZAP3; STR FRP; INC FRP
21BE 8B5818;	0436 GLO ZAP3; STR FRP; INC FRP
21C1 ;	0437
21C1 ;	0438 ..COMPUTE FREQUENCY INDEX OF MAXIMUM VALUE
21C1 ;	0439 ..SHOULD BE (MAX.ADDR - BASE.ADDR), IF POSITIVE
21C1 F8D9AC;	0440 LDI A.0(BASE+1); PLO ZAP1
21C4 8BF7AA2C;	0441 GLO ZAP3; SM; PLO ZAP2; DEC ZAP1
21C8 9B77BA;	0442 GHI ZAP3; SMB; PHI ZAP2
21CB C321D6;	0443 LBDF CMPFMX ..IF POSITIVE, THEN
21CE ;	0444 ..GOTO COMPUTE FMAX
21CE ;	0445
21CE ;	0446 ..ELSE NEGATIVE,
21CE ;	0447 ..INDEX = (YSLWA+1 - YSFWA) - DIFFERENCE
21CE 8AFC00AA;	0448 GLO ZAP2; ADI #00; PLO ZAP2
21D2 9A7C04BA;	0449 GHI ZAP2; ADCI #04; PHI ZAP2
21D6 ;	0450
21D6 ;	0451 CMPFMX: ..COMPUTE FMAX
21D6 ;	0452 ..FMAX = (INDEX/4) + 3.0, + 0.5 UNDERSTOOD
21D6 ;	0453 ..DIVIDE BY 4
21D6 9AF6BA;	0454 GHI ZAP2; SHR; PHI ZAP2
21D9 8A76AA;	0455 GLO ZAP2; SHRC; PLO ZAP2
21DC 9AF6BA;	0456 GHI ZAP2; SHR; PHI ZAP2
21DF 8A76;	0457 GLO ZAP2; SHRC
21E1 ;	0458
21E1 ;	0459 ..ADD 3 AND STORE AS FMAX
21E1 FC03AA;	0460 ADI #03; PLO ZAP2
21E4 9A7C005818;	0461 GHI ZAP2; ADCI #00; STR FRP; INC FRP
21E9 8A5818;	0462 GLO ZAP2; STR FRP; INC FRP
21EC ;	0463



```

21EC ;          0464 ..STORE VALUE AT MAXIMUM INTO MEMORY
21EC 4B5818;    0465          LDA ZAP3; STR FRP; INC FRP
21EF 0B5818;    0466          LDN ZAP3; STR FRP; INC FRP
21F2 ;          0467
21F2 ;          0468 ..STORE SHIFT VALUE OF MAXIMUM
21F2 ;          0469 ..LOAD TOTAL SHIFT, SUBTRACT 1 (INHERENT IN ORDER
21F2 ;          0470 ..ROUTINUE), THEN STORE TO MAXIMUM'S EXPONENT
21F2 F8DBAC;    0471          LDI A.0(TOTLSH+1); PLO ZAP1
21F5 0CFF01AA;  0472          LDN ZAP1; SMI #01; PLO ZAP2
21F9 2C0C;      0473          DEC ZAP1; LDN ZAP1
21FB 7F005818;  0474          SMBI #00; STR FRP; INC FRP
21FF 8A58;      0475          GLO ZAP2; STR FRP
2201 ;          0476
2201 ;          0477 ..THEN RETURN TO MAIN
2201 D5;         0478          EXIT
2202 C4C4C4;     0479          ,#C4C4C4
2205 ;          0480
2205 ;          0481
2205 ;          0482 ..*****
2205 ;          0483 ..SMBDCK.SR          SMOOTH BOUNDARY CHECK
2205 ;          0484 ..
2205 ;          0485 ..CALL THIS ROUTINUE TO CHECK INPUT OR OUTPUT PTR
2205 ;          0486 ..          FOR YSLWA+1 OR YSLWA+3 CONDITION
2205 ;          0487 ..THEN CHECK FOR DONE
2205 ;          0488 ..R(MA) IS THE INPUT PTR, R(MP) IS THE OUTPUT PTR
2205 ;          0489 ..R(ZAP1) & R(MQ.1) ARE USED IN THIS ROUTINUE
2205 ;          0490 ..*****
2205 ;          0491
2205 ;          0492
2205 ;          0493 ..RETURN PORTION
2205 ;          0494 EXSBCK:          ..EXIT SMOOTH BOUNDARY CHECK
2205 D3;         0495          SEP PC          ..RESET TO PC
2206 ;          0496
2206 ;          0497
2206 ;          0498 ..ENTER HERE FOR BOUNDARY CHECK
2206 ;          0499 SMBDCK:          ..SMOOTHED BOUNDARY CHECK
2206 ;          0500
2206 ;          0501 ..COMPARE INPUT POINTER FOR PAST END
2206 9DFF3E;     0502          GHI MA; SMI A.1(YSLWA+1)
2209 CB220F;     0503          LBNF *+06          ..IF WITHIN BOUNDARY, THEN
220C ;          0504          ..SKIP TO OUTPUT PTR CHECK
220C ;          0505
220C ;          0506 ..ELSE OFF END, RESET TO START
220C F83ABD;     0507          LDI A.1(YSFWA); PHI MA
220F ;          0508
220F ;          0509 ..COMPARE OUTPUT POINTER FOR PAST END
220F 99FF3E;     0510          GHI MP; SMI A.1(YSLWA+1)
2212 CB2218;     0511          LBNF *+06          ..IF WITHIN BOUNDARY, THEN
2215 ;          0512          ..SKIP TO DONE CHECK
2215 ;          0513

```

ASP FFT Program  
Program Code

2215 ;	0514 ..ELSE OFF END, RESET TO START
2215 F83AB9;	0515 LDI A.1(YSFWA); PHI MP
2218 ;	0516
2218 ;	0517 ..-----
2218 ;	0518 CKMVDN: ..CHECK MOVE DONE
2218 ;	0519 ..COUNT DOWN MOVE COUNT BY 4
2218 F8CBAC;	0520 LDI A.0(MOVECT+1); PLO ZAP1
221B EC;	0521 SEX ZAP1
221C F804F573BE;	0522 LDI #04; SD; STXD; PHI MQ
2221 F8007573;	0523 LDI #00; SDB; STXD
2225 ;	0524
2225 ;	0525 ..CHECK FOR DONE WITH MOVE
2225 ;	0526 ..CHECK HIGH BYTE, IF NOT #00
2225 ;	0527 ..RETURN WITH NOT DONE FLAG (NOT #00)
2225 CA2205;	0528 LBNZ EXSBCK
2228 ;	0529
2228 ;	0530 ..CHECK LOW BYTE AND RETURN WITH FLAG EITHER
2228 ;	0531 ..NOT #00, NOT DONE
2228 ;	0532 ..OR IS #00, IS DONE
2228 9EC02205;	0533 GHI MQ; LBR EXSBCK
222C ;	0534
222C ;	0535
222C ;	0536 ..#####
222C ;	0537 ..END OF SMOOTH & TAG MAXIMUM

A.2.2.3 CKFMAX (Check F0 Maximum)

```

0000 ;          0001
0000 ;          0002 ..CKFMAX.SR          25 MAY 80          12:35 PM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..CKFMAX.SR          CHECK FREQUENCY OF MAXIMUM
0000 ;          0244 ..
0000 ;          0245 ..FIRST,THIS ROUTINE MOVES TO THE OUTPUT LIST
0000 ;          0246 ..      THE LENGTH OF THE FFT (64, 128, 256 OR 512)
0000 ;          0247 ..THEN, IT CHECKS THE FREQUENCY INDEX OF THE MAXIMUM
0000 ;          0248 ..      VALUE, FMAX (POSSIBLY 3 TO (FFTLN/2)-8)
0000 ;          0249 ..      FOR BOUNDARY VALIDITY (NOT AT HEAD OR TAIL)
0000 ;          0250 ..      IF OK, THEN RETURN TO FFTMN WITH D = #00
0000 ;          0251 ..      ELSE, STORE FMAX AS F0
0000 ;          0252 ..      ZERO OTHER OUTPUT LOCATIONS
0000 ;          0253 ..      AND RETURN WITH D.NE.#00
0000 ;          0254 ..
0000 ;          0255 ..THE ROUTINE ALSO CORRECTS THE VALUE AT FMAX
0000 ;          0256 ..(MAXIMUM ENERGY VALUE) FOR:
0000 ;          0257 ..      1. INSTRUMENT RESPONSE (PER FREQUENCY INDEX)
0000 ;          0258 ..      2. SPECTRA LENGTH (DELETED HERE,
0000 ;          0259 ..      NOW DONE IN BOSS - 5/25/80)
0000 ;          0260 ..      3. CHANNEL GAIN
0000 ;          0261 ..#####
0000 ;          0262
0000 ;          0263
0000 ;          0264 ..EXTERNAL ROUTINES
0000 ;          0265
0000 ;          0266          SNMULT=#2804          ..SINE MULTIPLY
0000 ;          0267
0000 ;          0268 ..INTERNAL VARIABLES (FROM NOW ON)
0000 ;          0269
0000 ;          0270          GAIN=#3062          ..AMPLIFIER GAIN
0000 ;          0271
0000 ;          0272          MXADDR=CSVLU+2          ..ADDRESS OF MAXIMUM
0000 ;          0273          FMAX=MXADDR+2          ..FREQUENCY INDEX OF MAX
0000 ;          0274          MXVALU=FMAX+2          ..MAXIMUM VALUE (INTEGER)
0000 ;          0275          MXEXP=MXVALU+2          ..EXPONENT (SHIFT) VALUE
0000 ;          0276          ..OF MAX
0000 ;          0277

```

ASP FFT Program  
Program Code

```

0000 ;          0278          LNFFT=MXEXP+2          ..LENGTH OF FFT
0000 ;          0279          FO=LNFFT+2          ..CORNER FREQUENCY
0000 ;          0280          GMAINT=FO+2          ..GAMMA INTEGER
0000 ;          0281          GMAFRC=GMAINT+2          ..GAMMA FRACTION
0000 ;          0282          MXNINT=GMAFRC+2          ..MAXIMUM ENERGY INTEGER
0000 ;          0283          MXNEXP=MXNINT+2          ..MAXIMUM ENERGY EXPONENT
0000 ;          0284          ..(CORRECTED MAX VALUE,
0000 ;          0285          ..THEN LONG PERIOD LEVEL)
0000 ;          0286
0000 ;          0287          INSFIX=#2E00          ..INSTRUMENT CORRECTION TBL
0000 ;          0288
0000 ;          0289          ..BEGIN HERE
0000 ;          0290
0000 ;          0291          ORG #2250
2250 ;          0292
2250 ;          0293          ..*****
2250 ;          0294          CKFMAX:          ..CHECK FREQUENCY OF MAXIMUM
2250 ;          0295          ..FIRST PASS LENGTH OF FFT
2250 ;          0296          ..(DIVIDE BY 4 TO GET NUMBER OF SPECTRA COMPONENTS)
2250 F8D0A8; 0297          LDI A.0(FFTLN); PLO FRP
2253 F8E8AC; 0298          LDI A.0(LNFFT); PLO ZAP1
2256 48F6BA; 0299          LDA FRP; SHR; PHI ZAP2
2259 0876AA; 0300          LDN FRP; SHRC; PLO ZAP2
225C 9AF6; 0301          GHI ZAP2; SHR
225E BA5C1C; 0302          PHI ZAP2; STR ZAP1; INC ZAP1
2261 8A76; 0303          GLO ZAP2; SHRC
2263 AA5C1C; 0304          PLO ZAP2; STR ZAP1; INC ZAP1
2266 ; 0305
2266 ; 0306          ..NOW GET R(MP) TO CORRECT LENGTH'S BOUNDARY
2266 ; 0307          ..START WITH ASSUMED LENGTH = 256
2266 F823B9; 0308          LDI A.1(BD256+7); PHI MP
2269 F8B3A9; 0309          LDI A.0(BD256+7); PLO MP
226C E9; 0310          SEX MP
226D ; 0311
226D ; 0312          MBLNTH:          ..MAYBE LENGTH
226D ; 0313          ..TEST FOR LNFFT IN R(ZAP2) => LENGTH @ M(R(MP))
226D 8AF729; 0314          GLO ZAP2; SM; DEC MP
2270 9A7729; 0315          GHI ZAP2; SMB; DEC MP
2273 C32281; 0316          LBDF ENGCR T          ..IF LNFFT => LENGTH, THEN
2276 ; 0317          ..GOTO ENERGY CORRECTION
2276 ; 0318
2276 ; 0319          ..ELSE GET TO NEXT LENGTH
2276 89FF06A9; 0320          GLO MP; SMI #06; PLO MP
227A 997F00B9; 0321          GHI MP; SMI #00; PHI MP
227E ; 0322
227E ; 0323          ..AND RETEST
227E C0226D; 0324          LBR MBLNTH          ..GOTO MAYBE LENGTH
2281 ; 0325

```



```

2281 ;          0326 ..-----
2281 ;          0327 ENGCRT:          ..ENERGY CORRECTION
2281 ;          0328 ..FIRST CORRECT FMAX VALUE FOR INSTRUMENT RESPONSE
2281 ;          0329 ..GET FMAX INDEX VALUE INTO R(MQ)
2281 F8E2A8;      0330          LDI A.0(FMAX); PLO FRP
2284 48BE;        0331          LDA FRP; PHI MQ
2286 08AE;        0332          LDN FRP; PLO MQ
2288 ;          0333
2288 ;          0334 ..GET INSTRUMENT CORRECTION SKIP VALUE INTO R(ZAP3)
2288 09AB29;      0335          LDN MP; PLO ZAP3; DEC MP
228B ;          0336
228B ;          0337 ..CALL LENGTH FIX SUBROUTINE TO GET
228B ;          0338 ..CORRECT FIX TABLE VALUE (FMAX)
228B F823B0;      0339          LDI A.1(LNFX); PHI ZAP
228E F852A0;      0340          LDI A.0(LNFX); PLO ZAP
2291 D0;          0341          SEP ZAP          ..ZAP = LENGTH FIX SUBR.
2292 ;          0342
2292 ;          0343 ..GET R(MA) TO FMAX VALUE LOCATION
2292 F830BD;      0344          LDI A.1(MXVALU); PHI MA
2295 F8E4AD;      0345          LDI A.0(MXVALU); PLO MA
2298 ;          0346
2298 ;          0347 ..CALL SINE MULTIPLY FOR PRODUCT OF
2298 ;          0348 ..M(FMAX) * INSFIX(FMAX)
2298 F828B0;      0349          LDI A.1(SNMULT); PHI ZAP
229B F804A0;      0350          LDI A.0(SNMULT); PLO ZAP
229E D0;          0351          SEP ZAP          ..ZAP = SINE MULTIPLY
229F ;          0352
229F ;          0353 ..AT HALFWAY POINT
229F ;          0354 ..GET R(AC) & R(MA) TO MAX ENERGY INTEGER
229F F830BFBD;    0355          LDI A.1(MXNINT); PHI AC; PHI MA
22A3 F8F0AFAD;    0356          LDI A.0(MXNINT); PLO AC; PLO MA
22A7 ;          0357
22A7 ;          0358 ..THEN RETURN TO STORE RESULTS
22A7 D0;          0359          SEP ZAP
22A8 ;          0360
22A8 ;          0361 ..NOW YOU CORRECT THIS RESULT
22A8 ;          0362 ..TO NORMALIZE IT TO THE 1ST INDEX VALUE
22A8 ;          0363 ..(THE TABLE IS NORMALIZED TO THE 13TH INDEX VALUE
22A8 ;          0364 ..FOR DYNAMIC RANGE CONSIDERATIONS.
22A8 ;          0365 ..YOU NEED TO MULTIPLY BY INSFIX(13) / INSFIX(1).
22A8 ;          0366 ..SEE THE TABLE HEADER FOR FULL EXPLANATION.)
22A8 ;          0367
22A8 ;          0368 ..R(MA) IS AT MAX ENERGY INTEGER
22A8 ;          0369 ..GET R(MQ) TO THE TABLE NORMALIZE VALUE
22A8 F823BE;      0370          LDI A.1(FXNMFR); PHI MQ
22AB F8B8AE;      0371          LDI A.0(FXNMFR); PLO MQ
22AE ;          0372

```

ASP FFT Program  
Program Code

22AE ;	0373	..MULTIPLY MXNINT BY INSFIX NORMALIZE FRACTION
22AE DO;	0374	SEP ZAP ..ZAP = SINE MULTIPLY
22AF ;	0375	
22AF ;	0376	..AT HALFWAY GET R(AC) BACK TO MAX ENERGY INTEGER
22AF 2F2F;	0377	DEC AC; DEC AC
22B1 ;	0378	
22B1 ;	0379	..AND GET R(MA) TO MAXIMUM EXPONENT (FOR LATER)
22B1 F830BD;	0380	LDI A.1(MXEXP); PHI MA
22B4 F8E6AD;	0381	LDI A.0(MXEXP); PLO MA
22B7 ;	0382	
22B7 ;	0383	..RETURN TO STORE NORMALIZED RESULT
22B7 DO;	0384	SEP ZAP ..ZAP = MIDDLE OF SNMULT
22B8 ;	0385	
22B8 ;	0386	..THEN STORE "NORMALIZE FIX" EXPONENT AT MXNEXP
22B8 4E5F1F;	0387	LDA MQ; STR AC; INC AC
22BB OE5F;	0388	LDN MQ; STR AC
22BD ;	0389	
22BD ;	0390	.....
22BD ;	0391	..THIS SECTION WAS TO CORRECT FOR SPECTRA LENGTH
22BD ;	0392	..HOWEVER THE ALGORITHM IS WRONG -
22BD ;	0393	..CORRECT IS TO DIVIDE BY THE SAMPLE TIME.
22BD ;	0394	..THIS IS DONE IN THE BOSS.
22BD ;	0395	..MOST OF THE CODE REMAINS HOWEVER 5/25/80
22BD ;	0396	
22BD ;	0397	..LOAD EXPONENT (SHIFT) VALUE OF MAXIMUM @ M(R(MA))
22BD 4DBB4D;	0398	LDA MA; PHI ZAP3; LDA MA
22C0 ;	0399	
22C0 ;	0400	..SUBTRACT LENGTH CORRECTION AT M(R(MP))
22C0 E9C4AB29;	0401	SEX MP; NOP; PLO ZAP3; DEC MP
22C4 9BC4C4BB;	0402	GHI ZAP3; NOP; NOP; PHI ZAP3
22C8 ;	0403	
22C8 ;	0404	..ADD THAT VALUE TO MAX ENERGY EXPONENT
22C8 EF;	0405	SEX AC
22C9 8BF473;	0406	GLO ZAP3; ADD; STXD
22CC 9B745F;	0407	GHI ZAP3; ADC; STR AC
22CF ;	0408	
22CF ;	0409	..THEN GO FOR FULL SCALE CHECK OF MAX ENERGY INT/EXP
22CF F823B0;	0410	LDI A.1(FLSCCK); PHI ZAP
22D2 F86FA0;	0411	LDI A.0(FLSCCK); PLO ZAP
22D5 DO;	0412	SEP ZAP ..ZAP = FULL SCALE CHECK
22D6 ;	0413	

```

22D6 ; 0414 .....
22D6 ; 0415 ..NOW CORRECT FOR CHANNEL GAIN
22D6 ; 0416 ..MAX.ENERGY <- MAX.ENERGY/(G.PHONE GAIN * AMP GAIN)
22D6 ; 0417
22D6 ; 0418 ..FOR GEOPHONE GAIN
22D6 ; 0419 ..MAX.ENG.INT <- MAX.ENG.INT * (GPHPWR/G.PHONE GAIN)
22D6 ; 0420 .. WHERE: GPHPWR = 2 * * G.PHONE EXP
22D6 ; 0421 ..MAX.ENERGY.EXP <- MAX.ENERGY.EXP - G.PHONE EXP
22D6 ; 0422
22D6 ; 0423 ..GET R(MA) & R(AC) TO MAX.ENERGY.INT
22D6 F830BDBF; 0424 LDI A.1(MXNINT); PHI MA; PHI AC
22DA F8F0ADAF; 0425 LDI A.0(MXNINT); PLO MA; PLO AC
22DE ; 0426
22DE ; 0427 ..GET R(MQ) TO GEOPHONE GAIN FRACTION
22DE F823BE; 0428 LDI A.1(GPHFRC); PHI MQ
22E1 F8B4AE; 0429 LDI A.0(GPHFRC); PLO MQ
22E4 ; 0430
22E4 ; 0431 ..CALL SINE MULTIPLY FOR PRODUCT
22E4 F828B0; 0432 LDI A.1(SNMULT); PHI ZAP
22E7 F804A0; 0433 LDI A.0(SNMULT); PLO ZAP
22EA D0; 0434 SEP ZAP ..ZAP = SINE MULTIPLY
22EB ; 0435
22EB ; 0436 ..AT HALFWAY POINT ADD G.PHONE EXP
22EB 1DED; 0437 INC MA; SEX MA
22ED 1E0EF473; 0438 INC MQ; LDN MQ; ADD; STXD
22F1 2E0E745D; 0439 DEC MQ; LDN MQ; ADC; STR MA
22F5 ; 0440
22F5 ; 0441 ..THEN RETURN TO STORE MULTIPLICATION RESULT
22F5 D0; 0442 SEP ZAP ..ZAP = MIDDLE OF SNMULT
22F6 ; 0443
22F6 ; 0444 ..NOW CORRECT FOR AMP GAIN
22F6 ; 0445 ..AMP GAIN = 2 * * POWER
22F6 ; 0446 ..MAX.ENERGY.EXP <- MAX.ENERGY.EXP - POWER
22F6 ; 0447
22F6 ; 0448 ..GET GAIN INTO R(MP)
22F6 F862A8; 0449 LDI A.0(GAIN); PLO FRP
22F9 48BB; 0450 LDA FRP; PHI ZAP3
22FB 08AB; 0451 LDN FRP; PLO ZAP3
22FD ; 0452
22FD ; 0453 ..LOAD R(AC) WITH POWER = 0 (2 * * 0 = 1)
22FD F800BFAF; 0454 LDI #00; PHI AC; PLO AC
2301 ; 0455
2301 ; 0456 TSTGN: ..TEST GAIN
2301 ; 0457 ..DIVIDE WORKING GAIN BY 2, THEN TEST FOR #0000
2301 9BF6BB; 0458 GHI ZAP3; SHR; PHI ZAP3
2304 8B76AB; 0459 GLO ZAP3; SHRC; PLO ZAP3
2307 CA230E; 0460 LBNZ INCPWR ..IF LOW NOT #00, THEN
230A ; 0461 ..GOTO INCREMENT POWER
230A 9BC22312; 0462 GHI ZAP3; LBZ SBTRXP ..IF BOTH = #00, THEN
230E ; 0463 ..GOTO SUBTRACT EXP
230E ; 0464

```



ASP FFT Program  
Program Code

```

230E ;          0465 INCPWR:                ..INCREMENT POWER
230E ;          0466 ..(ELSE) INCREMENT POWER, THEN GOTO RETEST
230E 1FC02301;  0467                      INC AC; LBR TSTGN
2312 ;          0468
2312 ;          0469 SBTRXP:                ..SUBTRACT EXPONENT
2312 ;          0470 ..TAKE POWER AWAY FROM MAXIMUM ENERGY EXPONENT
2312 1DED;      0471                      INC MA; SEX MA
2314 8FF573;    0472                      GLO AC; SD; STXD
2317 9F755D;    0473                      GHI AC; SDB; STR MA
231A ;          0474
231A ;          0475 ..THEN RETEST FULL SCALE CHECK FOR MAX.INT/EXP
231A F823B0;    0476                      LDI A.1(FLSCCK); PHI ZAP
231D F86FA0;    0477                      LDI A.0(FLSCCK); PLO ZAP
2320 D0;        0478                      SEP ZAP                ..ZAP = FULL SCALE CHECK
2321 ;          0479
2321 ;          0480 ..-----
2321 ;          0481 ..SET UP OUTPUT VARIABLES FOR SENDING TO BOSS
2321 ;          0482 ..(IN CASE OF ABORTION)
2321 ;          0483
2321 ;          0484 ..GET FMAX VALUE INTO R(ZAP3)
2321 F8E2A8;    0485                      LDI A.0(FMAX); PLO FRP
2324 48BB;      0486                      LDA FRP; PHI ZAP3
2326 48AB;      0487                      LDA FRP; PLO ZAP3
2328 ;          0488
2328 ;          0489 ..STORE FMAX AT FO
2328 9B5C1C;    0490                      GHI ZAP3; STR ZAP1; INC ZAP1
232B 8B5C1C;    0491                      GLO ZAP3; STR ZAP1; INC ZAP1
232E ;          0492
232E ;          0493 ..ZERO GAMMA INTEGER (MAYBE NEEDED FOR ABORT)
232E F8005C1C;  0494                      LDI #00; STR ZAP1; INC ZAP1
2332 5C1C;      0495                      STR ZAP1; INC ZAP1
2334 ;          0496
2334 ;          0497 ..ZERO GAMMA EXPONENT
2334 5C1C5C;    0498                      STR ZAP1; INC ZAP1; STR ZAP1
2337 ;          0499
2337 ;          0500 ..-----
2337 ;          0501 TSTBDR:                ..TEST BOUNDARY
2337 ;          0502 ..CHECK FMAX IN R(ZAP3) <= TAIL BOUNDARY @ M(R(MP))
2337 E9;        0503                      SEX MP
2338 8BF529;    0504                      GLO ZAP3; SD; DEC MP
233B 9B7529;    0505                      GHI ZAP3; SDB; DEC MP
233E AF;        0506                      PLO AC
233F CB234E;    0507                      LBNF EXFMCK                ..IF FMAX > TLBDRY, THEN
2342 ;          0508                      ..GOTO EXIT WITH D.NE.#00
2342 ;          0509

```



```

2342 ;                0510 ..ELSE CHECK FMAX => HEAD BOUNDARY
2342 8BF729;          0511         GLO ZAP3; SM; DEC MP
2345 9B77;            0512         GHI ZAP3; SMB
2347 AF;              0513         PLO AC
2348 CB234E;          0514         LBNF EXFMCK          ..IF FMAX < HDBDRY, THEN
234B ;                0515                                ..GOTO EXIT WITH D.NE.#00
234B ;                0516
234B ;                0517 ..ELSE, FMAX OK
234B ;                0518 ..RETURN WITH D.EQ.#00
234B F800AF;          0519         LDI #00; PLO AC
234E ;                0520
234E ;                0521 ..-----
234E ;                0522 EXFMCK:          ..EXIT FMAX CHECK
234E D5;              0523         EXIT
234F C4C4C4;          0524         ,#C4C4C4
2352 ;                0525
2352 ;                0526 ..*****
2352 ;                0527 ..LNFIX.SR          ..LENGTH FIX SUBROUTINE
2352 ;                0528 ..
2352 ;                0529 ..THIS ROUTINE GETS A POINTER TO THE CORRECT
2352 ;                0530 ..LOCATION IN THE INSTRUMENT CORRECTION TABLE
2352 ;                0531 ..IT EXPECTS THE INDEX POINTER TO BE IN R(MQ)
2352 ;                0532 ..AND THE SKIP VALUE TO BE IN R(ZAP3.0)
2352 ;                0533 ..ON RETURN
2352 ;                0534 .. R(MQ) <- INSFIX + [R(MQ) * 2*(R(ZAP3.0)+1)]
2352 ;                0535 ..*****
2352 ;                0536
2352 ;                0537
2352 ;                0538 LNFIX:          ..LENGTH FIX SUBROUTINE
2352 ;                0539 ..DOUBLE INDEX ONCE (BECAUSE DOUBLE WORDS)
2352 8EFEAE;            0540         GLO MQ; SHL; PLO MQ
2355 9E7EBE;            0541         GHI MQ; SHLC; PHI MQ
2358 ;                0542
2358 ;                0543 MBMORE:          ..MAYBE MORE SKIPS
2358 ;                0544 ..CHECK FOR SKIP = #00
2358 8B;                0545         GLO ZAP3
2359 C22366;            0546         LBZ ADFXFW          ..IF #00, THEN
235C ;                0547                                ..GOTO ADD FIX TABLE FWA
235C ;                0548
235C ;                0549 ..ELSE DECREMENT SKIP COUNT
235C 2B;                0550         DEC ZAP3
235D ;                0551
235D ;                0552 ..AND DOUBLE INDEX VALUE
235D 8EFEAE;            0553         GLO MQ; SHL; PLO MQ
2360 9E7EBE;            0554         GHI MQ; SHLC; PHI MQ
2363 ;                0555
2363 ;                0556 ..THEN RECHECK FOR SKIP = #00
2363 C02358;            0557         LBR MBMORE
2366 ;                0558

```

ASP FFT Program  
Program Code

2366 ;	0559 ADFXFW:	..ADD FIX TABLE FWA
2366 ;	0560 ..ADD FIX FWA TO MODIFIED INDEX VALUE	
2366 8EFC00AE;	0561 GLO MQ; ADI A.0(INSFIX); PLO MQ	
236A 9E7C2EBE;	0562 GHI MQ; ADCI A.1(INSFIX); PHI MQ	
236E ;	0563	
236E ;	0564 ..THEN RETURN TO CALLING PROGRAM	
236E D3;	0565 SEP PC	..RESET TO PC
236F ;	0566	
236F ;	0567 ..*****	
236F ;	0568 ..FLSCCK.SR	..FULL SCALE CHECK
236F ;	0569 ..	
236F ;	0570 ..THIS ROUTINE CHECKS THAT MAXIMUM ENERGY INTEGER	
236F ;	0571 ..IS AT FULL SCALE AND ADJUSTS INTEGER & EXPONENT	
236F ;	0572 ..IF NECESSARY	
236F ;	0573 ..IT HAS NO INPUT REQUIREMENTS	
236F ;	0574 ..AND USES ONLY R(MQ) & R(AC)	
236F ;	0575 ..*****	
236F ;	0576	
236F ;	0577	
236F ;	0578 FLSGCK:	..FULL SCALE CHECK
236F ;	0579 ..GET R(AC) TO MAXIMUM ENERGY INTEGER	
236F F830BF;	0580 LDI A.1(MXNINT); PHI AC	
2372 F8F0AF;	0581 LDI A.0(MXNINT); PLO AC	
2375 EF;	0582 SEX AC	
2376 ;	0583	
2376 ;	0584 INTCHK:	..INTEGER CHECK
2376 ;	0585 ..CHECK FOR MSB (NOT SIGN BIT) = 1	
2376 F840F2;	0586 LDI #40; AND	
2379 CA2393;	0587 LBNZ EXFSCK	..IF = 1, THEN
237C ;	0588	..GOTO EXIT FULL SCALE CHECK
237C ;	0589	
237C ;	0590 ..ELSE LOAD, SHIFT UP ONCE, AND STORE	
237C 4FBE;	0591 LDA AC; PHI MQ	
237E 0FFE73;	0592 LDN AC; SHL; STXD	
2381 9E7E5F;	0593 GHI MQ; SHLC; STR AC	
2384 ;	0594	
2384 ;	0595 ..ALSO SUBTRACT 1 FROM EXPONENT	
2384 F8F3AF;	0596 LDI A.0(MXNEXP+1); PLO AC	
2387 0FFF0173;	0597 LDN AC; SMI #01; STXD	
238B 0F7F0073;	0598 LDN AC; SMBI #00; STXD	
238F ;	0599	
238F ;	0600 ..GET BACK TO INTEGER, AND RECHECK	
238F 2F;	0601 DEC AC	
2390 C02376;	0602 LBR INTCHK	
2393 ;	0603	
2393 ;	0604 EXFSCK:	..EXIT FULL SCALE CHECK
2393 ;	0605 ..RESET TO CALLING PROGRAM	
2393 D3;	0606 SEP PC	
2394 ;	0607	

ASP FFT Program  
Program Code

```

2394 ;          0608 ..*****
2394 ;          0609 ..TABLE OF BOUNDARIES & VALUES
2394 ;          0610
2394 0004;          0611 BD32:      ,#0004          ..HEAD BOUNDARY (1ST OK)
2396 000D;          0612      ,#000D  ..13          ..TAIL BOUNDARY (LAST OK)
2398 06;           0613      ,#06           ..LENGTH CORRECTION EXPONENT
2399 03;           0614      ,#03           ..CORRECTION TBL. SKIP VALUE
239A 0000;          0615      ,#0000          ..SPECTRA LENGTH (NORMALLY)
239C ;           0616
239C 0007;          0617 BD64:      ,#0007
239E 0021;          0618      ,#0021  ..33
23A0 07;           0619      ,#07
23A1 02;           0620      ,#02
23A2 0040;          0621      ,#0040
23A4 ;           0622
23A4 000D;          0623 BD128:    ,#000D  ..13
23A6 0051;          0624      ,#0051  ..81
23A8 08;           0625      ,#08
23A9 01;           0626      ,#01
23AA 0080;          0627      ,#0080
23AC ;           0628
23AC 001A;          0629 BD256:    ,#001A  ..26
23AE 0099;          0630      ,#0099  ..153
23B0 09;           0631      ,#09
23B1 00;           0632      ,#00
23B2 0100;          0633      ,#0100
23B4 ;           0634
23B4 ;           0635 GPHFRC:      ..GEOPHONE GAIN FRACTION
23B4 6F5C;          0636      ,#6F5C  .. .435 = .870 * 2**(-1)
23B6 ;           0637      ..= 28508/32768 * 2**(-1)
23B6 ;           0638      ..= ,#6F5C      * 2**(#FFFF)
23B6 ;           0639 GPHEXP:      ..GEOPHONE GAIN EXPONENT
23B6 FFFF;          0640      ,#FFFF
23B8 ;           0641
23B8 ;           0642 FXNMFR:      ..FIXTBL NORMALIZE FRACTION
23B8 45A2;          0643      ,#45A2  .. 3.40E-2 = .5440 * 2**(-4)
23BA ;           0644      ..= 17826/32768 * 2**(-4)
23BA ;           0645      ..= ,#45A2 * 2**(#FFFC)
23BA ;           0646 FXNMPX:      ..FIXTBL NORMALIZE EXPONENT
23BA FFFC;          0647      ,#FFFC
23BC ;           0648
23BC ;           0649 ..#####
23BC ;           0650 ..END OF CKFMAX.SR

```



ASP FFT Program  
Program Code

A.2.2.4 GMACPT (GAMMA Compute)

```

0000 ;          0001
0000 ;          0002 ..GMACPT.SR          25 MAY 80          3:00 PM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..GMACPT.SR          ..GAMMA COMPUTE
0000 ;          0244 ..
0000 ;          0245 ..THIS ROUTINUE COMPUTES
0000 ;          0246 ..
0000 ;          0247 ..GAMMA = [LOG(Y1) - LOG(Y2)] / [LOG(X2) - LOG(X1)]
0000 ;          0248 ..
0000 ;          0249 ..WITH: X1 = FMAX + X1OFFSET (FOR FMAX + 6HZ)
0000 ;          0250 ..      X2 = FMAX + X2OFFSET (FOR FMAX + 20HZ)
0000 ;          0251 ..      Y1 = AVGY(FMAX + Y1OFFSET)
0000 ;          0252 ..      Y2 = AVGY(FMAX + Y2OFFSET)
0000 ;          0253 ..
0000 ;          0254 ..WHERE: OFFSETS ARE A FUNCTION OF SPECTRA LENGTH
0000 ;          0255 ..      AVGY(N) = [(INSFIX(N')*SM(N)) + ...
0000 ;          0256 ..      (INSFIX(N+3') * SM(N+3))] / 4
0000 ;          0257 ..WITH: N' = N * (256 / SPECTRA.LENGTH)
0000 ;          0258 ..      (TO ACCOUNT FOR DIFFERENT RESOLUTIONS PER
0000 ;          0259 ..      SPECTRA.LENGTH)
0000 ;          0260 ..
0000 ;          0261 ..THE ONLY OUTPUTS ARE GAMMA INTEGER (GMAINT)
0000 ;          0262 ..      AND GAMMA FRACTION (GMAFRC)
0000 ;          0263 ..EITHER GOOD VALUES (POSITIVE < MAXIMUM)
0000 ;          0264 ..OR ERROR FLAG
0000 ;          0265 ..      #FFFF.FFFFF FOR NEGATIVE GAMMA
0000 ;          0266 ..#####
0000 ;          0267
0000 ;          0268
0000 ;          0269 ..EXTERNAL ROUTINUES
0000 ;          0270
0000 ;          0271          LNFI#=#2352          ..LENGTH FIX
0000 ;          0272          FLSCCK=#2308          ..FULL SCALE CHECK
0000 ;          0273          SNMULT=#2804          ..SINE MULTIPLY
0000 ;          0274          DIV=#10B7          ..RCA 32 BIT BY 16 BIT
0000 ;          0275          ..DIVIDE
0000 ;          0276

```



```

0000 ;      0277 ..INTERNAL VARIABLES (FROM NOW ON)
0000 ;      0278
0000 ;      0279      MXADDR=CSVLU+2      ..ADDRESS OF MAXIMUM
0000 ;      0280      FMAX=MXADDR+2      ..FREQUENCY INDEX OF MAX
0000 ;      0281      MXVALU=FMAX+2      ..MAXIMUM VALUE (INTEGER)
0000 ;      0282      MXEXP=MXVALU+2     ..EXPONENT (SHIFT) VALUE
0000 ;      0283      ..OF MAX
0000 ;      0284
0000 ;      0285      LNFFT=MXEXP+2      ..LENGTH OF FFT
0000 ;      0286      FO=LNFFT+2        ..CORNER FREQUENCY
0000 ;      0287      GMAINT=FO+2        ..GAMMA INTEGER
0000 ;      0288      GMAFRC=GMAINT+2    ..GAMMA FRACTION
0000 ;      0289      MXNINT=GMAFRC+2    ..MAXIMUM ENERGY INTEGER
0000 ;      0290      MXNEXP=MXNINT+2    ..MAXIMUM ENERGY EXPONENT
0000 ;      0291      ..(CORRECTED MAX VALUE,
0000 ;      0292      ..THEN LONG PERIOD LEVEL)
0000 ;      0293
0000 ;      0294      NUMLOG=MXNEXP+2     ..NUMERATOR (DIFFERENCE) LOG
0000 ;      0295      ..8 BIT MAN. & 16 BIT CHAR.
0000 ;      0296      DENLOG=NUMLOG+3    ..DENOMINATOR (DIF.) LOG
0000 ;      0297      ..8 BIT MAN. & 16 BIT CHAR.
0000 ;      0298      YXWORK=DENLOG+3    ..WORKING Y OR X SCRATCHPAD
0000 ;      0299      ..(TEMPORARY)
0000 ;      0300
0000 ;      0301 ..BEGIN HERE
0000 ;      0302
0000 ;      0303      ORG #23F0
0000 ;      0304
0000 ;      0305 ..*****
0000 ;      0306 GMACPT:      ..GAMMA COMPUTE
0000 ;      0307 ..GET R(MP) TO CORRECT LENGTH'S TABLE
0000 ;      0308 ..GET LNFFT INTO R(ZAP2)
0000 ;      0309      LDI A.0(LNFFT); PLO FRP
0000 ;      0310      LDA FRP; PHI ZAP2
0000 ;      0311      LDA FRP; PLO ZAP2
0000 ;      0312
0000 ;      0313 ..GET R(MP) TO ASSUMED LENGTH 256
0000 ;      0314      LDI A.1(TBL256+13); PHI MP
0000 ;      0315      LDI A.0(TBL256+13); PLO MP
0000 ;      0316      SEX MP
0000 ;      0317
0000 ;      0318 MBLNTH:      ..MAYBE LENGTH
0000 ;      0319 ..TEST FOR LNFFT IN R(ZAP2) => LENGTH @ M(R(MP))
0000 ;      0320      GLO ZAP2; SM; DEC MP
0000 ;      0321      GHI ZAP2; SMB; DEC MP
0000 ;      0322      LBDF GOY1      ..IF LNFFT => LENGTH, THEN
0000 ;      0323      ..GOTO GO FOR Y1
0000 ;      0324

```

ASP FFT Program  
Program Code

2407 ;	0325 ..ELSE GET TO NEXT LENGTH
2407 89FF0CA9;	0326 GLO MP; SMI #0C; PLO MP
240B 997F00B9;	0327 GHI MP; SMBI #00; PHI MP
240F ;	0328
240F ;	0329 ..AND RETEST
240F C023FE;	0330 LBR MBLNTH ..GOTO MAYBE LENGTH
2412 ;	0331
2412 ;	0332 ..*****
2412 ;	0333 GOY1: ..GOING FOR Y1 VALUE
2412 ;	0334 ..SET FLAG FOR FIRST PASS
2412 F8FFBA;	0335 LDI #FF; PHI ZAP2
2415 ;	0336
2415 ;	0337 ..-----
2415 ;	0338 LYINIT: ..LOG2(AVG(Y1)) INITIALIZE
2415 ;	0339 ..GET ADDRESS OF FIRST Y IN GROUP INTO R(MA)
2415 ;	0340 ..GET ADDRESS OF FMAX INTO R(MA)
2415 F8E0A8;	0341 LDI A.0(MXADDR); PLO FRP
2418 48BD48;	0342 LDA FRP; PHI MA; LDA FRP
241B ;	0343
241B ;	0344 ..ADD Y ADDRESS OFFSET
241B E9;	0345 SEX MP
241C F4AD29;	0346 ADD; PLO MA; DEC MP
241F 9D74BD29;	0347 GHI MA; ADC; PHI MA; DEC MP
2423 ;	0348
2423 ;	0349 ..GET ADDRESS OF INSFIX VALUE INTO R(MQ)
2423 ;	0350 ..GET FMAX INDEX INTO R(MQ)
2423 48BE48;	0351 LDA FRP; PHI MQ; LDA FRP
2426 ;	0352
2426 ;	0353 ..ADD Y INDEX OFFSET
2426 F4AE29;	0354 ADD; PLO MQ; DEC MP
2429 9E7C00BE;	0355 GHI MQ; ADCI #00; PHI MQ
242D ;	0356
242D ;	0357 ..LOAD INSFIX TABLE SKIP VALUE INTO R(ZAP3.0)
242D 09AB29;	0358 LDN MP; PLO ZAP3; DEC MP
2430 ;	0359
2430 ;	0360 ..CALL LENGTH FIX TO GET TO INSFIX ENTRY
2430 F823B0;	0361 LDI A.1(LNFIIX); PHI ZAP
2433 F852A0;	0362 LDI A.0(LNFIIX); PLO ZAP
2436 D0;	0363 SEP ZAP ..ZAP = LENGTH FIX
2437 ;	0364

ASP FFT Program  
Program Code

```

2437 ; 0365 .....
2437 ; 0366 ..YOU'RE NOW AT THE FIRST OF 4 ENTRIES TO AVG/LOG
2437 ; 0367 ..SET SERIES COUNTER IN R(ZAP2.0) TO 4
2437 F804AA; 0368 LDI #04; PLO ZAP2
243A ; 0369
243A ; 0370 ..GET R(FRP) TO 24 BITS OF WORKING Y, AND ZERO
243A F8FAA8; 0371 LDI A.0(YKWORK); PLO FRP
243D F8005818; 0372 LDI #00; STR FRP; INC FRP
2441 5818; 0373 STR FRP; INC FRP
2443 58; 0374 STR FRP
2444 ; 0375
2444 ; 0376 ..GET R(AC) TO FRAM
2444 F830BF; 0377 LDI A.1(FRAM); PHI AC
2447 ; 0378
2447 ; 0379 ..GET R(ZAP) TO SINE MULTIPLY
2447 F828B0; 0380 LDI A.1(SNMULT); PHI ZAP
244A F804A0; 0381 LDI A.0(SNMULT); PLO ZAP
244D ; 0382
244D ; 0383 .....
244D ; 0384 MLTADD: ..MULTIPLY, THEN ADD
244D ; 0385 ..CHECK FOR YSLWA+1 (MIGHT BE BAD AT START)
244D 9DFF3E; 0386 GHI MA; SMI A.1(YSLWA+1)
2450 CB2456; 0387 LBNF MLVLCR ..IF < BOUNDARY, THEN GOTO
2453 ; 0388 ..MULT. VALUE * CORRECTION
2453 ; 0389
2453 ; 0390 ..ELSE > BOUNDARY, RESET POINTER TO START OF YSFIFO
2453 F83ABD; 0391 LDI A.1(YSFWA); PHI MA
2456 ; 0392
2456 ; 0393 MLVLCR: ..MULT. VALUE BY CORRECTION
2456 ; 0394 ..CALL SINE MULTIPLY FOR CORRECTED Y
2456 D0; 0395 SEP ZAP ..ZAP = SINE MULTIPLY
2457 ; 0396
2457 ; 0397 ..AT HALFWAY, GET R(MA) TO NEXT SMOOTHED VALUE
2457 ; 0398 ..(EVERY OTHER ONE IS #0000)
2457 1D1D; 0399 INC MA; INC MA
2459 ; 0400
2459 ; 0401 ..GET R(MQ) TO NEXT INSFIX
2459 E9; 0402 SEX MP
245A 8EF4AE; 0403 GLO MQ; ADD; PLO MQ
245D 9E7C00BE; 0404 GHI MQ; ADCI #00; PHI MQ
2461 ; 0405
2461 ; 0406 ..GET R(AC) TO GARBAGE (#30FD)
2461 F8FDAF; 0407 LDI #FD; PLO AC
2464 ; 0408
2464 ; 0409 ..THEN RETURN TO GET PRODUCT
2464 D0; 0410 SEP ZAP ..ZAP = MIDDLE OF SNMULT
2465 ; 0411

```



ASP FFT Program  
Program Code

2465 ;	0412 ..ADD PRODUCT TO WORKING Y
2465 E8;	0413 SEX FRP
2466 8BF473;	0414 GLO ZAP3; ADD; STXD
2469 9B7473;	0415 GHI ZAP3; ADC; STXD
246C F8007458;	0416 LDI #00; ADC; STR FRP
2470 1818;	0417 INC FRP; INC FRP
2472 ;	0418
2472 ;	0419 ..DECREMENT SERIES COUNTER, CHECK FOR #00
2472 2A8A;	0420 DEC ZAP2; GLO ZAP2
2474 CA244D;	0421 LBNZ MLTADD ..IF NOT #00, THEN
2477 ;	0422 ..GOTO MULT/ADD NEXT Y
2477 ;	0423
2477 ;	0424 ..*****
2477 ;	0425 ..ELSE DONE WITH 4 Y'S, DIVIDE BY 4 FOR AVERAGE
2477 2828;	0426 DEC FRP; DEC FRP
2479 08F65818;	0427 LDN FRP; SHR; STR FRP; INC FRP
247D 08765818;	0428 LDN FRP; SHRC; STR FRP; INC FRP
2481 087658;	0429 LDN FRP; SHRC; STR FRP ..DIVIDE BY 2 ONCE
2484 ;	0430
2484 2828;	0431 DEC FRP; DEC FRP
2486 08F65818;	0432 LDN FRP; SHR; STR FRP; INC FRP
248A 08765818;	0433 LDN FRP; SHRC; STR FRP; INC FRP
248E 087658;	0434 LDN FRP; SHRC; STR FRP ..DIVIDE BY 2 AGAIN
2491 ;	0435
2491 ;	0436 ..GET R(FRP) TO HIGH BYTE OF RESULT
2491 28;	0437 DEC FRP
2492 ;	0438
2492 ;	0439 ..CALL LOG2 FOR LOG2(AVG(Y))
2492 F825B0;	0440 LDI A.1(LOG2); PHI ZAP
2495 F8D0A0;	0441 LDI A.0(LOG2); PLO ZAP
2498 D0;	0442 SEP ZAP ..ZAP = LOG2(M(R(FRP)))
2499 ;	0443
2499 ;	0444 ..RETURNS WITH CHAR. IN R(MQ.0), MANTISSA IN R(AC)
2499 ;	0445
2499 ;	0446 ..ON RETURN, CHECK 2ND PASS FLAG
2499 9A;	0447 GHI ZAP2
249A C224AE;	0448 LBZ SBT2ND ..IF 2ND PASS, THEN
249D ;	0449 ..GOTO SUBTRACT 2ND
249D ;	0450
249D ;	0451 ..ELSE 1ST PASS, STORE RESULT AT NUMERATOR'S LOG
249D F8F6A8;	0452 LDI A.0(NUMLOG+2); PLO FRP
24A0 E8;	0453 SEX FRP
24A1 8F739F73;	0454 GLO AC; STXD; GHI AC; STXD
24A5 8E73;	0455 GLO MQ; STXD
24A7 ;	0456
24A7 ;	0457 ..GET R(MP) TO Y2 ADDRESS OFFSET
24A7 29;	0458 DEC MP
24A8 ;	0459
24A8 ;	0460 ..SET 2ND PASS FLAG
24A8 F800BA;	0461 LDI #00; PHI ZAP2
24AB ;	0462



ASP FFT Program  
Program Code

```

24AB ;          0463 ..GOTO LOG2(AVG(Y)) INITIALIZE
24AB C02415;    0464          LBR LYINIT
24AE ;          0465
24AE ;          0466 ..-----
24AE ;          0467 SBT2ND:          ..SUBTRACT LOG2(AVG(Y2))
24AE ;          0468 ..SUBTRACT Y1.MAN (@ NUM.MAN) - Y2.MAN AND STORE
24AE F8F6A8;    0469          LDI A.0(NUMLOG+2); PLO FRP
24B1 E8;        0470          SEX FRP
24B2 8FF573;    0471          GLO AC; SD; STXD
24B5 9F7573;    0472          GHI AC; SDB; STXD
24B8 ;          0473
24B8 ;          0474 ..CHECK FOR MANTISSA POSITIVE
24B8 C324C0;    0475          LBDF STNCHR          ..IF MANTISSA POSITIVE, THEN
24BB ;          0476                      ..GOTO SUBTRACT NUM.CHAR
24BB ;          0477
24BB ;          0478 ..ELSE MANTISSA NEGATIVE, BORROW FROM CHARACTERISTIC
24BB ;          0479 ..(SUBTRACT ONE MORE, LATER)
24BB 1E;        0480          INC MQ
24BC ;          0481
24BC ;          0482 ..AND ADD #8000 (POSITIVE 32768) TO MANTISSA
24BC ;          0483 ..(WILL FLIP SIGN BIT)
24BC 18FC8073;  0484          INC FRP; ADI #80; STXD
24C0 ;          0485
24C0 ;          0486 STNCHR:          ..SUBTRACT NUM.CHAR
24C0 ;          0487 ..SUBTRACT Y1.CHAR (@NUM.CHAR) - Y2.CHAR AND STORE
24C0 8EF573;    0488          GLO MQ; SD; STXD
24C3 ;          0489
24C3 ;          0490 ..CHECK FOR POSITIVE CHARACTERISTIC
24C3 C324D2;    0491          LBDF DENCPT          ..IF POSITIVE, THEN
24C6 ;          0492                      ..GOTO DENOMINATOR COMPUTE
24C6 ;          0493
24C6 ;          0494 ..ELSE NEGATIVE CHARACTERISTIC (ERROR)
24C6 ;          0495 ..STORE #FFFF AT GAMMA INTEGER & FRACTION
24C6 F8EFA8;    0496          LDI A.0(GMAFRC+1); PLO FRP
24C9 F8FF7373;  0497          LDI #FF; STXD; STXD
24CD 7373;      0498          STXD; STXD
24CF ;          0499
24CF ;          0500 ..THEN EXIT WITH THAT ERROR FLAG
24CF C02567;    0501          LBR EXGMCT          ..GOTO EXIT GAMMA COMPUTE
24D2 ;          0502

```

ASP FFT Program  
Program Code

```

24D2 ; 0503 ..*****
24D2 ; 0504 DENCPT: ..DENOMINATOR COMPUTE
24D2 ; 0505 ..COMPUTE LOG2(X2) - LOG2(X1)
24D2 ; 0506
24D2 ; 0507 ..GET FMAX INDEX INTO R(MQ)
24D2 F8E2A8; 0508 LDI A.0(FMAX); PLO FRP
24D5 48BE08AE; 0509 LDA FRP; PHI MQ; LDN FRP; PLO MQ
24D9 ; 0510
24D9 ; 0511 ..ADD X2 INDEX OFFSET AND STORE AT WORKING X
24D9 E929; 0512 SEX MP; DEC MP
24DB F8FCA8; 0513 LDI A.0(YXWORK+2); PLO FRP
24DE 8EF45828; 0514 GLO MQ; ADD; STR FRP; DEC FRP
24E2 9E7C0058; 0515 GHI MQ; ADCI #00; STR FRP
24E6 ; 0516
24E6 ; 0517 ..CALL LOG2 FOR LOG2(X2)
24E6 D0; 0518 SEP ZAP ..ZAP = LOG2(M(R(FRP)))
24E7 ; 0519
24E7 ; 0520 ..ON RETURN, STORE LOG2(X2) AT DENOMINATOR'S LOG
24E7 F8F9A8; 0521 LDI A.0(DENLOG+2); PLO FRP
24EA E8; 0522 SEX FRP
24EB 8F739F73; 0523 GLO AC; STXD; GHI AC; STXD
24EF 8E73; 0524 GLO MQ; STXD
24F1 ; 0525
24F1 ; 0526 ..SUBTRACT (X2 TO X1 INDEX OFFSET) FROM X2 INDEX
24F1 F8FCA8; 0527 LDI A.0(YXWORK+2); PLO FRP
24F4 2909F573; 0528 DEC MP; LDN MP; SD; STXD
24F8 F8007558; 0529 LDI #00; SDB; STR FRP
24FC ; 0530
24FC ; 0531 ..CALL LOG2 AGAIN FOR LOG2(X1)
24FC D0; 0532 SEP ZAP ..ZAP = LOG2(M(R(FRP)))
24FD ; 0533
24FD ; 0534 ..ON RETURN
24FD ; 0535 ..SUBTRACT X2.MAN (@ DEN.MAN) - X1.MAN AND STORE
24FD F8F9A8; 0536 LDI A.0(DENLOG+2); PLO FRP
2500 E8; 0537 SEX FRP
2501 8FF573; 0538 GLO AC; SD; STXD
2504 9F7573; 0539 GHI AC; SDB; STXD
2507 ; 0540
2507 ; 0541 ..CHECK FOR MANTISSA POSITIVE
2507 C3250F; 0542 LBDF STDCHR ..IF MANTISSA POSITIVE, THEN
250A ; 0543 ..GOTO SUBTRACT DEN.CHAR
250A ; 0544
250A ; 0545 ..ELSE MANTISSA NEGATIVE, BORROW FROM CHARACTERISTIC
250A ; 0546 ..(SUBTRACT ONE MORE, LATER)
250A 1E; 0547 INC MQ
250B ; 0548
250B ; 0549 ..AND ADD #8000 (POSITIVE 32768) TO MANTISSA
250B ; 0550 ..(WILL FLIP SIGN BIT)
250B 18FC8073; 0551 INC FRP; ADI #80; STXD
250F ; 0552

```

```

250F ;          0553 STDCHR:          ..SUBTRACT DEN.CHAR
250F ;          0554 ..SUBTRACT X2.CHAR (@DEN.CHAR) - X1.CHAR AND STORE
250F 8EF573;    0555          GLO MQ; SD; STXD
2512 ;          0556
2512 ;          0557 ..NOTE: ASSURED ANSWER POSITIVE > #0000
2512 ;          0558
2512 ;          0559 ..*****
2512 ;          0560 ..DIVIDE NUM-CHAR.MANTISSA / DEN-CHAR.MANTISSA
2512 ;          0561 ..TO DIVIDE YOU NEED TO SCALE THEM BOTH DOWN
2512 ;          0562 ..UNTIL CHARACTERISTICS ARE #00
2512 ;          0563 ..THEN DIVIDE THE "MANTISSAS"
2512 ;          0564
2512 ;          0565 CHRCHK:          ..CHARACTERISTIC CHECK
2512 ;          0566 ..GET NUMERATOR CHARACTERISTIC FOR CHECK
2512 F8F4A8;    0567          LDI A.0(NUMLOG); PLO FRP
2515 48;        0568          LDA FRP
2516 ;          0569
2516 ;          0570 ..GET TO DENOMINATOR CHAR. FOR CHECK
2516 1818;      0571          INC FRP; INC FRP
2518 ;          0572
2518 ;          0573 ..CHECK NUMERATOR CHAR.
2518 CA251F;    0574          LBNZ SHFLOG          ..IF.NE.#00, THEN
251B ;          0575          ..GOTO SHIFT LOG
251B ;          0576
251B ;          0577 ..CHECK DENOMINATOR CHAR.
251B 08;        0578          LDN FRP
251C C2253A;    0579          LBZ DVNMDN          ..IF BOTH = #00, THEN
251F ;          0580          ..GOTO DIVIDE NUM/DEN
251F ;          0581
251F ;          0582 SHFLOG:          ..SHIFT LOG
251F ;          0583 ..BOTH.NE.#00, YOU HAVE TO SHIFT DOWN
251F ;          0584 ..GET BACK TO NUMERATOR CHAR.
251F 282828;    0585          DEC FRP; DEC FRP; DEC FRP
2522 ;          0586
2522 ;          0587 SHTHLG:          ..SHIFT THIS LOG
2522 ;          0588 ..SHIFT LOG DOWN BY 1
2522 08F65818;  0589          LDN FRP; SHR; STR FRP; INC FRP
2526 08C7;      0590          LDN FRP; LSNF          ..IF NO SHIFT FROM CHAR.,
2528 F980;      0591          ORI #80          ..THEN SKIP BIT SET
252A F65818;    0592          SHR; STR FRP; INC FRP
252D 08765818;  0593          LDN FRP; SHRC; STR FRP; INC FRP
2531 ;          0594
2531 ;          0595 ..TEST FOR 2ND PASS (WAS IT DENOM.LOG ?)
2531 88FDF7;    0596          GLO FRP; SDI A.0(DENLOG)
2534 CB2512;    0597          LBNF CHRCHK          ..IF 2ND PASS, THEN
2537 ;          0598          ..GO BACK TO CHAR. RECHECK
2537 ;          0599
2537 ;          0600 ..ELSE (WAS 1ST, NUMER.LOG), GOTO SHIFT DENOM.LOG
2537 C02522;    0601          LBR SHTHLG          ..GOTO SHIFT THIS LOG
253A ;          0602          ..@ M(R(FRP)), DENOM.LOG
253A ;          0603

```



ASP FFT Program  
Program Code

253A ;	0604 ..-----
253A ;	0605 DVNMDN: ..DIVIDE NUM/DEN
253A ;	0606 ..WANT TO DIVIDE NUMERATOR BY DENOMINATOR
253A ;	0607 ..AND PUT ANSWER INTO GAMMA INTEGER
253A ;	0608
253A ;	0609 ..GET R(MA) TO NUMERATOR
253A F830BD;	0610 LDI A.1(NUMLOG+1); PHI MA
253D F8F5AD;	0611 LDI A.0(NUMLOG+1); PLO MA
2540 ;	0612
2540 ;	0613 ..LOAD NUMERATOR INTO LOW DIVIDEND R(AC)
2540 4DBF4DAF;	0614 LDA MA; PHI AC; LDA MA; PLO AC
2544 ;	0615
2544 ;	0616 ..ZERO HIGH 16 BITS OF DIVIDEND, R(MQ)
2544 F800BEAE;	0617 LDI #00; PHI MQ; PLO MQ
2548 ;	0618
2548 ;	0619 ..GET R(MA) TO DENOMINATOR
2548 1D;	0620 INC MA
2549 ;	0621
2549 ;	0622 ..CALL RCA DIVIDE FOR
2549 ;	0623 ..32 BIT IN R(MQ.AC) / 16 BIT IN M(R(MA))
2549 D410B7;	0624 CALL DIV
254C ;	0625
254C ;	0626 ..ON RETURN STORE QUOTIENT AT GAMMA INTEGER
254C F8ECA8;	0627 LDI A.0(GMAINT); PLO FRP
254F 9F5818;	0628 GHI AC; STR FRP; INC FRP
2552 8F5818;	0629 GLO AC; STR FRP; INC FRP
2555 ;	0630
2555 ;	0631 ..REMAINDER IS IN R(MQ)
2555 ;	0632 ..SHIFT OVER TO SAVE ROOM FOR THE FRACTION'S SIGN
2555 9EF6BE;	0633 GHI MQ; SHR; PHI MQ
2558 8E76AE;	0634 GLO MQ; SHRC; PLO MQ
255B ;	0635
255B ;	0636 ..ZERO LOWER 16 INPUT BITS (FOR ANOTHER DIVIDE)
255B F800BFAF;	0637 LDI #00; PHI AC; PLO AC
255F ;	0638
255F ;	0639 ..RECALL RCA DIVIDE
255F D410B7;	0640 CALL DIV
2562 ;	0641
2562 ;	0642 ..YOU KNOW RESULTS GOOD (NO OVERFLOW)
2562 ;	0643 ..STORE (FRACTION) QUOTIENT AT GAMMA FRACTION
2562 9F5818;	0644 GHI AC; STR FRP; INC FRP
2565 8F58;	0645 GLO AC; STR FRP
2567 ;	0646
2567 ;	0647 ..FALL THROUGH TO EXIT BACK TO FFTMN
2567 ;	0648
2567 ;	0649 ..*****
2567 ;	0650 EXGMCT: ..EXIT GAMMA COMPUTE
2567 ;	0651 ..RETURN TO CALLING PROGRAM
2567 D5;	0652 EXIT
2568 C4C4C4;	0653 ,#C4C4C4
256B ;	0654



```

256B ; 0655 ..#####
256B ; 0656 ..LOG2.SR          ..LOG2(M(R(FRP)))
256B ; 0657 ..
256B ; 0658 ..THIS ROUTINUE FINDS THE LOG, BASE 2,
256B ; 0659 ..OF THE NUMBER POINTED TO BY R(FRP)
256B ; 0660 ..(REQUIRED TO BE POSITIVE; IT CHECKS FOR #0000)
256B ; 0661 ..
256B ; 0662 ..FIRST, IT SHIFTS THE VALUE UP TO MAXIMUM SCALE
256B ; 0663 ..      (THE NUMBER OF SHIFTS = THE CHARACTERISTIC)
256B ; 0664 ..THEN, THE MANTISSA IS OBTAINED FROM A LOOKUP
256B ; 0665 ..      TABLE OF 128 LOG VALUES USING THE "MAXED"
256B ; 0666 ..      VALUE AS THE ADDRESS POINTER
256B ; 0667 ..
256B ; 0668 ..ON OUTPUT THE CHARACTERISTIC IS IN R(MQ.0) AND
256B ; 0669 ..      THE MANTISSA IS IN R(AC)
256B ; 0670 ..#####
256B ; 0671
256B ; 0672 ..LOCATE HERE
256B ; 0673
256B ; 0674          ORG #25CF
25CF ; 0675
25CF ; 0676 ..LOCAL VARIABLES
25CF ; 0677
25CF ; 0678          LGBASE=#2D00          ..LOG TABLE @ #2D00-#2DFE
25CF ; 0679
25CF ; 0680 ..*****
25CF ; 0681 EXLOG2:          ..EXIT LOG2
25CF D3; 0682          SEP PC          ..RESET TO PC
25D0 ; 0683
25D0 ; 0684 ..-----
25D0 ; 0685 ..ENTER HERE
25D0 ; 0686
25D0 ; 0687 LOG2:          ..LOG2(M(FRP))
25D0 ; 0688 ..LOAD VALUE TO BE LOGED INTO R(AC)
25D0 48BF; 0689          LDA FRP; PHI AC
25D2 08AF; 0690          LDN FRP; PLO AC
25D4 ; 0691
25D4 ; 0692 ..CHECK NUMBER FOR #0000
25D4 CA25DF; 0693          LBNZ LDMXCH
25D7 9FCA25DF; 0694          GHI AC; LBNZ LDMXCH ..IF.NE.#0000, THEN
25DB ; 0695          ..GOTO LOAD MAX CHAR.
25DB ; 0696
25DB ; 0697 ..ELSE RETURN WITH LOG2(#0000) = #00.#0000
25DB AE; 0698          PLO MQ
25DC C025CF; 0699          LBR EXLOG2          ..GOTO EXIT LOG2
25DF ; 0700
25DF ; 0701 LDMXCH:          ..LOAD MAXIMUM CHAR.
25DF ; 0702 ..LOAD 15 INTO CHAR IN R(MQ.0)
25DF ; 0703 ..(MAXIMUM CHAR. BECAUSE 2**15 = 32768)
25DF F80FAE; 0704          LDI #0F; PLO MQ
25E2 ; 0705

```

ASP FFT Program  
Program Code

25E2 ;	0706 UPCKSN:	..SHIFT UP, AND CHECK SIGN
25E2 ;	0707 ..DECREMENT CHAR COUNTER	
25E2 2E;	0708 DEC MQ	
25E3 ;	0709	
25E3 ;	0710 ..SHIFT NUMBER UP ONCE AND CHECK SIGN FOR MAX	
25E3 8FFEAF;	0711 GLO AC; SHL; PLO AC	
25E6 9F7EBF;	0712 GHI AC; SHLC; PHI AC	
25E9 FA80;	0713 ANI #80	
25EB C225E2;	0714 LBZ UPCKSN	..IF NOT AT MAX, THEN
25EE ;	0715	..GOTO RE(UP CHECK SIGN)
25EE ;	0716	
25EE ;	0717 ..ELSE YOU'RE AT MAXIMUM; USE THE HIGH 7 SIGNIFICANT	
25EE ;	0718 ..BITS FOR LOWER BYTE'S INDEX OFF BASE	
25EE ;	0719 ..(EX. LOG BASE =#2D00	
25EE ;	0720 .. NUMBER'S LOG @ 0010,1110,ABCD,EFG0)	
25EE ;	0721	
25EE ;	0722 ..GET R(AC) TO ADDRESS	
25EE 9FFEAF;	0723 GHI AC; SHL; PLO AC	
25F1 F82DBF;	0724 LDI A.1(LGBASE); PHI AC	
25F4 ;	0725	
25F4 ;	0726 ..LOAD MANTISSA INTO R(AC)	
25F4 4FBEOFAF;	0727 LDA AC; PHI MQ; LDN AC; PLO AC	
25F8 9EBF;	0728 GHI MQ; PHI AC	
25FA ;	0729	
25FA ;	0730 ..THEN RETURN THROUGH TOP	
25FA C025CF;	0731 LBR EXLOG2	
25FD ;	0732	
25FD ;	0733 ..#####	
25FD ;	0734 ..VALUES AND TABLES	
25FD ;	0735	
25FD ;	0736 LGSCFC:	..LOG SCALE FACTOR
25FD 2688;	0737 ,#2688	..3010
25FF ;	0738	
25FF 08;	0739 TBL32: ,#08 ..(12-4)	..(X1 TO X2) INDEX OFFSET
2600 0C;	0740 ,#0C ..12	..X2 INDEX OFFSET
2601 0E;	0741 ,#0E ..(2**4)-2	..INSFIX (RUNNING) SKIP
2602 03;	0742 ,#03	..INSFIX (INPUT) SKIP
2603 0B;	0743 ,#0B ..(12-1)	..Y2 INDEX OFFSET
2604 002C;	0744 ,#002C ..(4*11)	..Y2 ADDRESS OFFSET
2606 0E;	0745 ,#0E	..INSFIX (RUNNING) SKIP
2607 03;	0746 ,#03	..INSFIX (INPUT) SKIP
2608 03;	0747 ,#03 ..(4-1)	..Y1 INDEX OFFSET
2609 000C;	0748 ,#000C ..(4*3)	..Y1 ADDRESS OFFSET
260B 0000;	0749 ,#0000	..SPECTRA LENGTH (USUALLY)
260D ;	0750	

ASP FFT Program  
Program Code

```

260D 10;          0751 TBL64: ,#10    ..(24-8)
260E 18;          0752          ,#18    ..24
260F 06;          0753          ,#06    ..(2**3)-2
2610 02;          0754          ,#02
2611 17;          0755          ,#17    ..(24-1)
2612 005C;        0756          ,#005C  ..(4*23)
2614 06;          0757          ,#06
2615 02;          0758          ,#02
2616 07;          0759          ,#07    ..(8-1)
2617 001C;        0760          ,#001C  ..(4*7)
2619 0040;        0761          ,#0040
261B ;           0762
261B 20;          0763 TBL128: ,#20    ..(48-16)
261C 30;          0764          ,#30    ..48
261D 02;          0765          ,#02    ..(2**2)-2
261E 01;          0766          ,#01
261F 2F;          0767          ,#2F    ..(48-1)
2620 00BC;        0768          ,#00BC  ..(4*47)
2622 02;          0769          ,#02
2623 01;          0770          ,#01
2624 0F;          0771          ,#0F    ..(16-1)
2625 003C;        0772          ,#003C  ..(4*15)
2627 0080;        0773          ,#0080
2629 ;           0774
2629 40;          0775 TBL256: ,#40    ..(96-32)
262A 60;          0776          ,#60    ..96
262B 00;          0777          ,#00    ..(2**1)-2
262C 00;          0778          ,#00
262D 5F;          0779          ,#5F    ..(96-1)
262E 017C;        0780          ,#017C  ..(4*95)
2630 00;          0781          ,#00
2631 00;          0782          ,#00
2632 1F;          0783          ,#1F    ..(32-1)
2633 007C;        0784          ,#007C  ..(4*31)
2635 0100;        0785          ,#0100
2637 ;           0786
2637 ;           0787  ..#####
2637 ;           0788  ..END OF GMACPT.SR

```



ASP FFT Program  
Program Code

A.2.2.5 FOLPL (FOLPL Compute)

```

0000 ;          0001
0000 ;          0002 ..FOLPL.SR          26 FEB 80          6:00 PM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..FOLPL.SR          ..COMPUTE FO & LPL
0000 ;          0244 ..
0000 ;          0245 ..THIS ROUTINUE DOES TWO FUNCTIONS:
0000 ;          0246 ..
0000 ;          0247 ..FIRST, IT COMPUTES FO BY CHECKING GAMMA:
0000 ;          0248 ..IF          GAMMA < 1.0, THEN ABORT W/ WEIRD GAMMA
0000 ;          0249 ..IF 1.0 <= GAMMA < 1.5, THEN FO = 2.06 * FMAX.5
0000 ;          0250 ..IF 1.5 <= GAMMA < 2.5, THEN FO = 1.38 * FMAX.5
0000 ;          0251 ..IF 2.5 <= GAMMA < 3.5, THEN FO = 1.27 * FMAX.5
0000 ;          0252 ..IF 3.5 <= GAMMA < 4.5, THEN FO = 1.20 * FMAX.5
0000 ;          0253 ..IF 4.5 <= GAMMA < 5.5, THEN FO = 1.16 * FMAX.5
0000 ;          0254 ..IF 5.5 <= GAMMA < 6.5, THEN FO = 1.13 * FMAX.5
0000 ;          0255 ..IF 6.5 <= GAMMA < 7.5, THEN FO = 1.10 * FMAX.5
0000 ;          0256 ..IF 7.5 <= GAMMA,      THEN FO = 1.00 * FMAX.5
0000 ;          0257 ..
0000 ;          0258 ..THEN, IT COMPUTES THE LONG PERIOD LEVEL (LPL) BY:
0000 ;          0259 ..          LPL = MAXIMUM VALUE / 0.9
0000 ;          0260 ..          = 1.1 * MAXIMUM VALUE
0000 ;          0261 ..WHERE: MAXIMUM VALUE = ENERGY @ FMAX,
0000 ;          0262 ..          (ALREADY CORRECTED FOR INSTRUMENT RESPONCE,
0000 ;          0263 ..          GAIN FACTOR AND SPECTRA LENGTH)
0000 ;          0264 ..
0000 ;          0265 ..THE ONLY OUTPUTS ARE FO AND LPL-INTEGGER & EXPONENT
0000 ;          0266 ..(UNLESS ABORTED, THEN LEFT UNCHANGED)
0000 ;          0267 ..#####
0000 ;          0268
0000 ;          0269
0000 ;          0270 ..EXTERNAL ROUTINUES
0000 ;          0271
0000 ;          0272          SNMULT=#2804          ..SINE MULTIPLY
0000 ;          0273          ..DIVIDE
0000 ;          0274
0000 ;          0275 ..INTERNAL VARIABLES (FROM NOW ON)
0000 ;          0276
0000 ;          0277          MXADDR=CSVLU+2          ..ADDRESS OF MAXIMUM
0000 ;          0278          FMAX=MXADDR+2          ..FREQUENCY INDEX OF MAX
0000 ;          0279          MXVALU=FMAX+2          ..MAXIMUM VALUE (INTEGER)
0000 ;          0280          MXEXP=MXVALU+2          ..EXPONENT (SHIFT) VALUE
0000 ;          0281          ..OF MAX
0000 ;          0282

```



ASP FFT Program  
Program Code

```

0000 ;      0283      LNFFT=MXEXP+2      ..LENGTH OF FFT
0000 ;      0284      FO=LNFFT+2      ..CORNER FREQUENCY
0000 ;      0285      GMAINT=FO+2      ..GAMMA INTEGER
0000 ;      0286      GMAFRC=GMAINT+2      ..GAMMA FRACTION
0000 ;      0287      MXNINT=GMAFRC+2      ..MAXIMUM ENERGY INTEGER
0000 ;      0288      MXNEXP=MXNINT+2      ..MAXIMUM ENERGY EXPONENT
0000 ;      0289      ..(CORRECTED MAX VALUE,
0000 ;      0290      ..THEN LONG PERIOD LEVEL)
0000 ;      0291
0000 ;      0292      NUMLOG=MXNEXP+2      ..NUMERATOR (DIFFERENCE) LOG
0000 ;      0293      ..8 BIT MAN. & 16 BIT CHAR.
0000 ;      0294      DENLOG=NUMLOG+3      ..DENOMINATOR (DIF.) LOG
0000 ;      0295      ..8 BIT MAN. & 16 BIT CHAR.
0000 ;      0296
0000 ;      0297      ..BEGIN HERE
0000 ;      0298
0000 ;      0299      ORG #2C00
2C00 ;      0300
2C00 ;      0301      ..*****
2C00 ;      0302      FOLPL:      ..FO/LPL COMPUTE
2C00 ;      0303      ..INITIALIZE POINTERS
2C00 ;      0304      ..GET R(ZAP) SET UP FOR GAMMA COMPARE SUBROUTINE
2C00 F82CB0;      0305      LDI A.1(GMACPR); PHI ZAP
2C03 F8AAA0;      0306      LDI A.0(GMACPR); PLO ZAP
2C06 ;      0307
2C06 ;      0308      ..GET R(MA) TO FO (FMAX STORED THERE PREVIOUSLY)
2C06 F830BD;      0309      LDI A.1(FO); PHI MA
2C09 F8EAAD;      0310      LDI A.0(FO); PLO MA
2C0C ;      0311
2C0C ;      0312      ..GET R(AC) TO GARBAGE LOCATION (FOR SNMULT OUTPUT)
2C0C F830BF;      0313      LDI A.1(FRAM); PHI AC
2C0F F8FDAF;      0314      LDI #FD; PLO AC
2C12 ;      0315
2C12 ;      0316      ..-----
2C12 ;      0317      ..CHECK FOR GAMMA < 1.00
2C12 ;      0318      ..(JUST TEST INTEGER SINCE ASSURED FRACTION IS POS.)
2C12 F8EDA8;      0319      LDI A.0(GMAINT+1); PLO FRP
2C15 E8;      0320      SEX FRP
2C16 F801F528;      0321      LDI #01; SD; DEC FRP
2C1A F80075;      0322      LDI #00; SDB
2C1D CB2CA5;      0323      LBNF XFOLPL      ..IF GAMMA < 1.00, THEN
2C20 ;      0324      ..GOTO EXIT FO/LPL COMPUTE
2C20 ;      0325      ..(WITH WEIRD GAMMA AS FLAG)
2C20 ;      0326

```

ASP FFT Program  
Program Code

```

2C20 ; 0327 ..-----
2C20 ; 0328 ..ELSE CHECK FOR GAMMA < 1.5
2C20 ; 0329 ..R(FRP) IS AT GAMMA INTEGER
2C20 ; 0330 ..GET R(MQ) TO (1.0 TO 1.5) BOUNDARY
2C20 F82CBE; 0331 LDI A.1(BD1015+2); PHI MQ
2C23 F8D4AE; 0332 LDI A.0(BD1015+2); PLO MQ
2C26 ; 0333
2C26 ; 0334 ..CALL GAMMA COMPARE TO TEST FOR GAMMA < 1.5
2C26 D0; 0335 SEP ZAP ..ZAP = GAMMA COMPARE
2C27 ; 0336
2C27 ; 0337 ..ON RETURN IF DF = 1, THEN GAMMA => 1.5
2C27 C32C50; 0338 LBDF TBLTST ..IF GAMMA => 1.5, THEN
2C2A ; 0339 ..GOTO TABLE TEST
2C2A ; 0340
2C2A ; 0341 .....
2C2A ; 0342 ..ELSE GAMMA < 1.5
2C2A ; 0343 ..FO = 2.06 * FMAX.5
2C2A ; 0344 .. = FMAX + FMAX + (0.6 * FMAX) + 1.03
2C2A ; 0345
2C2A ; 0346 ..GET R(MQ) TO FACTOR FOR (1.00 TO 1.50) BOUNDARY
2C2A 2E2E; 0347 DEC MQ; DEC MQ
2C2C ; 0348
2C2C ; 0349 ..CALL SINE MULTIPLY FOR (0.6 * FMAX)
2C2C ; 0350 ..(R(MA) @ FO W/FMAX THERE AND R(AC) @ GARBAGE)
2C2C F828B0; 0351 LDI A.1(SNMULT); PHI ZAP
2C2F F804A0; 0352 LDI A.0(SNMULT); PLO ZAP
2C32 D0; 0353 SEP ZAP ..ZAP = SINE MULTIPLY
2C33 ; 0354
2C33 ; 0355 ..AT HALFWAY ADD FMAX + FMAX + 1
2C33 2DED; 0356 DEC MA; SEX MA
2C35 0DF473; 0357 LDN MA; ADD; STXD
2C38 0D745D1D; 0358 LDN MA; ADC; STR MA; INC MA
2C3C F801F473; 0359 LDI #01; ADD; STXD
2C40 F800745D1D; 0360 LDI #00; ADC; STR MA; INC MA
2C45 ; 0361
2C45 ; 0362 ..RETURN TO GET MULTIPLICATION RESULT
2C45 D0; 0363 SEP ZAP ..ZAP = MIDDLE OF SNMULT
2C46 ; 0364
2C46 ; 0365 ..ADD PRODUCT IN R(ZAP3) TO FO
2C46 ED; 0366 SEX MA
2C47 8BF473; 0367 GLO ZAP3; ADD; STXD
2C4A 9B745D; 0368 GHI ZAP3; ADC; STR MA
2C4D ; 0369
2C4D ; 0370 ..THEN GOTO COMPUTE LPL
2C4D C02C73; 0371 LBR LPLCPT
2C50 ; 0372

```

ASP FFT Program  
Program Code

```

2C50 ; 0373 ..*****
2C50 ; 0374 ..MAIN COMPARE SECTION
2C50 ; 0375 ..ON ENTERING YOU KNOW GAMMA => 1.50 (IT'S LEGAL)
2C50 ; 0376 ..NOW JUST FIND OUT WHERE
2C50 ; 0377
2C50 ; 0378 TBLTST: ..TABLE TEST
2C50 ; 0379 ..GET R(MQ) TABLE POINTER TO NEXT BOUNDARY
2C50 2E2E2E; 0380 DEC MQ; DEC MQ; DEC MQ
2C53 ; 0381
2C53 ; 0382 ..CALL GAMMA COMPARE
2C53 ; 0383 ..TO TEST FOR GAMMA => M(R(MQ)) + 0.50
2C53 D0; 0384 SEP ZAP ..ZAP = GAMMA COMPARE
2C54 ; 0385
2C54 ; 0386 ..ON RETURN CHECK FOR NOT => BOUNDARY
2C54 CB2C50; 0387 LBNF TBLTST ..IF GAMMA < BOUNDARY, THEN
2C57 ; 0388 ..GOTO NEXT TABLE ENTRY
2C57 ; 0389
2C57 ; 0390 ..ELSE YOU'VE GOT A HIT
2C57 ; 0391 ..GET R(MQ) TO MULTIPLICATION FACTOR
2C57 2E2E; 0392 DEC MQ; DEC MQ
2C59 ; 0393
2C59 ; 0394 ..CALL SINE MULTIPLY FOR
2C59 ; 0395 ..FACTOR @ M(R(MQ)-2) * FO @ M(R(MA))
2C59 ; 0396 ..(R(MA) @ FO W/FMAX THERE AND R(AC) @ GARBAGE)
2C59 F828B0; 0397 LDI A.1(SNMULT); PHI ZAP
2C5C F804A0; 0398 LDI A.0(SNMULT); PLO ZAP
2C5F D0; 0399 SEP ZAP ..ZAP = SINE MULTIPLY
2C60 ; 0400
2C60 ; 0401 ..AT HALFWAY ADD 1 TO FMAX
2C60 ; 0402 ..(THE RESULT OF THE 0.5 ON FMAX * FACTOR)
2C60 2D; 0403 DEC MA
2C61 0DFC015D2D; 0404 LDN MA; ADI #01; STR MA; DEC MA
2C66 0D7C005D1D; 0405 LDN MA; ADCI #00; STR MA; INC MA
2C6B ; 0406
2C6B ; 0407 ..THEN RETURN TO GET MULTIPLICATION RESULT
2C6B D0; 0408 SEP ZAP ..ZAP = MIDDLE OF SNMULT
2C6C ; 0409
2C6C ; 0410 ..ADD PRODUCT IN R(ZAP3) TO FO
2C6C ED; 0411 SEX MA
2C6D 8BF473; 0412 GLO ZAP3; ADD; STXD
2C70 9B745D; 0413 GHI ZAP3; ADC; STR MA
2C73 ; 0414
2C73 ; 0415 ..AND FALL THROUGH TO COMPUTE LPL
2C73 ; 0416

```



ASP FFT Program  
Program Code

```

2C73 ; 0417 ..*****
2C73 ; 0418 LPLCPT: ..LPL COMPUTE
2C73 ; 0419 ..HERE MULTIPLY MAX.ENERGY.INT (VALUE AT FMAX),
2C73 ; 0420 ..BY 1.10, AND ADJUST EXPONENT IF NECESSARY
2C73 ; 0421
2C73 ; 0422 ..GET R(MA) TO MAXIMUM ENERGY INTEGER
2C73 F830BD; 0423 LDI A.1(MXNINT); PHI MA
2C76 F8F0AD; 0424 LDI A.0(MXNINT); PLO MA
2C79 ; 0425
2C79 ; 0426 ..GET R(MQ) TO LPL FACTOR (0.10 = #OCCD)
2C79 F82CBE; 0427 LDI A.1(LPLFCT); PHI MQ
2C7C F8D5AE; 0428 LDI A.0(LPLFCT); PLO MQ
2C7F ; 0429
2C7F ; 0430 ..CALL SINE MULTIPLY FOR PRODUCT
2C7F DO; 0431 SEP ZAP ..ZAP = SINE MULTIPLY
2C80 ; 0432
2C80 ; 0433 ..AT HALFWAY GET R(MA) BACK TO MAX.INT.0
2C80 2D; 0434 DEC MA
2C81 ; 0435
2C81 ; 0436 ..GET R(AC) BACK TO OLD GARBAGE
2C81 F830BF; 0437 LDI A.1(FRAM); PHI AC
2C84 F8FD4F; 0438 LDI #FD; PLO AC
2C87 ; 0439
2C87 ; 0440 ..THEN RETURN TO GET MULTIPLICATION RESULT
2C87 DO; 0441 SEP ZAP ..ZAP = MIDDLE OF SNMULT
2C88 ; 0442
2C88 ; 0443 ..ADD RESULT TO MAXIMUM INTEGER
2C88 ED; 0444 SEX MA
2C89 8BF473; 0445 GLO ZAP3; ADD; STXD
2C8C 9B745D; 0446 GHI ZAP3; ADC; STR MA
2C8F ; 0447
2C8F ; 0448 ..CHECK FOR OVERFLOW INTO SIGN
2C8F FA80; 0449 ANI #80
2C91 C22CA5; 0450 LBZ XFOLPL ..IF NO OVERFLOW, THEN
2C94 ; 0451 ..GOTO EXIT F0/LPL COMPUTE
2C94 ; 0452
2C94 ; 0453 ..ELSE SHIFT MAXIMUM ENERGY INTEGER DOWN
2C94 0DF65D1D; 0454 LDN MA; SHR; STR MA; INC MA
2C98 0D765D; 0455 LDN MA; SHRC; STR MA
2C9B ; 0456
2C9B ; 0457 ..AND INCREMENT MAXIMUM ENERGY EXPONENT
2C9B 1D1D; 0458 INC MA; INC MA
2C9D 0DFC0173; 0459 LDN MA; ADI #01; STXD
2CA1 0D7C005D; 0460 LDN MA; ADCI #00; STR MA
2CA5 ; 0461
2CA5 ; 0462 ..THEN FALL THROUGH TO EXIT
2CA5 ; 0463

```



```

2CA5 ;          0464 ..*****
2CA5 ;          0465 XFOLPL:                ..EXIT F0/LPL COMPUTE
2CA5 ;          0466 ..RETURN TO CALLING PROGRAM
2CA5 D5;        0467             EXIT
2CA6 C4C4C4;    0468             ,#C4C4C4
2CA9 ;          0469
2CA9 ;          0470 ..#####
2CA9 ;          0471 ..GMACMP.SR                ..GAMMA COMPARE
2CA9 ;          0472 ..
2CA9 ;          0473 ..THIS ROUTINUE CHECKS FOR GAMMA < M(R(MQ)) + 0.50
2CA9 ;          0474 ..OR:
2CA9 ;          0475 ..      GMAINT.GMAFRC < M(R(MQ)) + 0.50 (#WXYZ.#4000)
2CA9 ;          0476 ..
2CA9 ;          0477 ..(NOTE: ASSURED BOTH INTEGER & FRACTION POSITIVE)
2CA9 ;          0478 ..
2CA9 ;          0479 ..AND RETURNS WITH:
2CA9 ;          0480 ..      DF = 0 IF GAMMA < M(R(MA)) + 0.50
2CA9 ;          0481 ..      DF = 1 IF GAMMA => M(R(MQ)) + 0.50
2CA9 ;          0482 ..
2CA9 ;          0483 ..IT ASSUMES R(FRP) POINTS TO GMAINT.1
2CA9 ;          0484 ..#####
2CA9 ;          0485
2CA9 ;          0486 EXGMCP:                ..EXIT GAMMA COMPUTE
2CA9 ;          0487 ..RETURN TO MAIN PROGRAM
2CA9 D3;        0488             SEP PC                ..RESET TO PC
2CAA ;          0489
2CAA ;          0490 ..-----
2CAA ;          0491 ..ENTER HERE
2CAA ;          0492
2CAA ;          0493 GMACPR:                ..GAMMA COMPARE
2CAA ;          0494 ..CHECK FOR GAMMA < M(R(MQ)) + 0.50
2CAA ;          0495 ..SET X TO R(FRP)
2CAA E8;        0496             SEX FRP
2CAB ;          0497
2CAB ;          0498 ..SUBTRACT 0.50 (#4000) FROM GAMMA FRACTION
2CAB 1818;      0499             INC FRP; INC FRP
2CAD F840F528; 0500             LDI #40; SD; DEC FRP
2CB1 ;          0501
2CB1 ;          0502 ..IF RESULT OF FRACTION SUBTRACTION IS POSITIVE
2CB1 ;          0503 ..SUBTRACT M(R(MQ)) FROM INTEGER
2CB1 OE;        0504             LDN MQ
2CB2 CF;        0505             LSDF                ..IF FRCT. SUBTRACTION NEG.
2CB3 FC01;      0506             ADI #01                ..SUBTRACT M(R(MQ)) + 1
2CB5 ;          0507
2CB5 ;          0508 ..NOW SUBTRACT INTEGER
2CB5 F528;      0509             SD; DEC FRP
2CB7 F80075;    0510             LDI #00; SDB
2CBA ;          0511
2CBA ;          0512 ..THEN EXIT THROUGH TOP WITH DF FLAG
2CBA C02CA9;    0513             LBR EXGMCP
2CBD ;          0514

```

ASP FFT Program  
Program Code

2CBD ;	0515	..#####
2CBD ;	0516	..TABLE OF VALUES
2CBD ;	0517	
2CBD ;	0518	BD1525: ..(1.50 TO 2.50) BOUNDARY
2CBD 30A4;	0519	,#30A4 ..38/100 ..ADDED (PRODUCT FACTOR)
2CBF 01;	0520	,#01 ..GAMMA BOUNDARY
2CC0 ;	0521	BD2535:
2CC0 228F;	0522	,#228F ..27/100
2CC2 02;	0523	,#02
2CC3 ;	0524	BD3545:
2CC3 199A;	0525	,#199A ..20/100
2CC5 03;	0526	,#03
2CC6 ;	0527	BD4555:
2CC6 147B;	0528	,#147B ..16/100
2CC8 04;	0529	,#04
2CC9 ;	0530	BD5565:
2CC9 10A4;	0531	,#10A4 ..13/100
2CCB 05;	0532	,#05
2CCC ;	0533	BD6575:
2CCC 0CCD;	0534	,#0CCD ..10/100
2CCE 06;	0535	,#06
2CCF ;	0536	BDGE75:
2CCF 0000;	0537	,#0000 ..00/100
2CD1 07;	0538	,#07
2CD2 ;	0539	
2CD2 ;	0540	BD1015: ..(1.00 TO 1.50) BOUNDARY
2CD2 07AE;	0541	,#07AE ..06/100
2CD4 01;	0542	,#01
2CD5 ;	0543	
2CD5 ;	0544	LPLFCT: ..LPL FACTOR
2CD5 0CCD;	0545	,#0CCD ..10/100
2CD7 ;	0546	
2CD7 ;	0547	..#####
2CD7 ;	0548	..END OF FO/LPL COMPUTE.SR

### A.2.3 Subroutines

#### A.2.3.1 SUBROT (Subroutine Block)

```

0000 ;          0001
0000 ;          0002 ..SUBROT.SR          21 JAN 80          2:10 PM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..SUBROT.SR          SUBROUTINE BLOCK
0000 ;          0244 ..#####
0000 ;          0245
0000 ;          0246
0000 ;          0247 ..VECTORS TO INTERNAL ROUTINES
0000 ;          0248
0000 ;          0249          ORG #2800
2800 3014;          0250          BR WRPTST          ..WRAPAROUND TEST
2802 3036;          0251          BR CSMULT          ..COSINE MULTIPLY
2804 304A;          0252          BR SNMULT          ..COSINE MULTIPLY
2806 305B;          0253          BR MLTMDL          ..MIDDLE OF FFT MULT
2808 306A;          0254          BR ADDSTF          ..ADD & STUFF
280A 3084;          0255          BR NEWVAL          ..NEW VALUE
280C 30AE;          0256          BR TRGSET          ..TRIG SET
280E C028F7;        0257          LBR SHPSET          ..SHAPE SET
2811 ;          0258
2811 ;          0259

```

ASP FFT Program  
Program Code

```

2811 ; 0260 ..#####
2811 ; 0261 ..WRPTST.SR WRAPAROUND TEST
2811 ; 0262 ..
2811 ; 0263 ..TESTS R(MA) FOR => BOUNDARY
2811 ; 0264 .. AND REINITIALIZES IF NECESSARY
2811 ; 0265 ..IF REINITIALIZED THE NEW ADDRESS WILL BE AS FAR
2811 ; 0266 .. INTO THE START OF THE TABLE AS IT HAD BEEN
2811 ; 0267 .. PAST THE END
2811 ; 0268 ..
2811 ; 0269 ..ASSUMES R(ZAP1) POINTS TO END BOUNDARY ADDRESS
2811 ; 0270 ..AND M(R(ZAP1) + 2) IS THE START BOUNDARY ADDRESS
2811 ; 0271 ..R(ZAP3.1) WILL BE A FLAG IN THE SUBROUTINE
2811 ; 0272 ..#####
2811 ; 0273
2811 ; 0274
2811 ; 0275
2811 ; 0276 ..RETURN PORTION
2811 ; 0277 CHGRN: ..CHANGED R(ZAP1) RETURN
2811 2C; 0278 DEC ZAP1
2812 2C; 0279 DEC ZAP1
2813 ; 0280 EXWRTS: ..EXIT WRAPAROUND TEST
2813 D3; 0281 SEP PC ..RESET TO PC
2814 ; 0282
2814 ; 0283 ..TEST PORTION - SUBTRACT BOUNDARY FROM POINTER
2814 ; 0284 WRPTST: ..WRAPAROUND TEST
2814 1C; 0285 INC ZAP1 ..GET TO LOW BOUNDARY
2815 EC; 0286 SEX ZAP1
2816 8D; 0287 GLO MA ..GET LOW POINTER
2817 F7; 0288 SM ..R(MA) - M(R(ZAP1)) -> D
2818 AD; 0289 PLO MA
2819 2C; 0290 DEC ZAP1 ..HIGH TOO
281A 9D; 0291 GHI MA
281B 77; 0292 SMB
281C BD; 0293 PHI MA
281D ; 0294
281D ; 0295 ..IF NEGATIVE RESULT, YOU'RE NOT AT BOUNDARY
281D 3B26; 0296 BNF UNDRBY ..BRANCH TO UNDER BDRY
281F ; 0297
281F ; 0298 ..IF POSITIVE RESULT ( => 0 )
281F ; 0299 ..MOVE BOUNDARY TO FRONT (FWA)
281F 1C; 0300 INC ZAP1
2820 1C; 0301 INC ZAP1
2821 ; 0302 ..SET FLAG FOR ADDING POSITIVE TO FRONT
2821 F8FF; 0303 LDI #FF
2823 BB; 0304 PHI ZAP3
2824 ; 0305 ..AND GO TO DOUBLE PRECISION ADD
2824 3029; 0306 BR DPRADD
2826 ; 0307

```



ASP FFT Program  
Program Code

2826 ;	0308 ..LESS THAN BOUNDARY
2826 ;	0309 ..SET FLAG FOR ADDING NEGATIVE TO REAR (LWA)
2826 F800;	0310 UNDRBY: LDI #00
2828 BB;	0311 PHI ZAP3
2829 ;	0312
2829 ;	0313 ..DOUBLE PRECISION ADD
2829 1C;	0314 DPRADD: INC ZAP1
282A 8D;	0315 GLO MA
282B F4;	0316 ADD
282C AD;	0317 PLO MA
282D 2C;	0318 DEC ZAP1
282E 9D;	0319 GHI MA
282F 74;	0320 ADC
2830 BD;	0321 PHI MA
2831 ;	0322
2831 ;	0323 ..CHECK IF ZAP1 AT END BOUNDARY (LWA)
2831 9B;	0324 GHI ZAP3
2832 3213;	0325 BZ EXWRTS
2834 ;	0326
2834 ;	0327 ..MUST BE AT FRONT, RESET TO END
2834 3011;	0328 BR CHGRTN
2836 ;	0329
2836 ;	0330
2836 ;	0331
2836 ;	0332

ASP FFT Program  
Program Code

```

2836 ; 0333 ..#####
2836 ; 0334 ..CS/SNMULT COSINE/SINE MULTIPLY
2836 ; 0335 ..
2836 ; 0336 ..THIS PROGRAM CAN BE ENTERED AT TWO PLACES
2836 ; 0337 ..COSINE MULTIPLY TAKES OPERAND B FROM M(CSVLU)
2836 ; 0338 ..SINE MULTIPLY TAKES OPERAND B FROM M(R(MQ))
2836 ; 0339 ..IN EITHER CASE, OPERAND A COMES FROM M(R(MA))
2836 ; 0340 ..
2836 ; 0341 ..THE RESULT GOES TO M(R(AC)) AND ALSO IS SAVED
2836 ; 0342 .. IN R(ZAP3)
2836 ; 0343 ..
2836 ; 0344 ..IT DELIVERS THE OPERANDS TO THE MULTIPLIER
2836 ; 0345 .. JUMPS OUT WHILE THE HARDWARE OPERATES
2836 ; 0346 .. THEN RETURNS TO STORE THE RESULT
2836 ; 0347 ..SINCE ONE OF THE OPERANDS IS A FRACTION
2836 ; 0348 .. (TRIG VALUE) YOU HAVE TO MULTIPLY
2836 ; 0349 .. RESULT BY 2 (SHIFT LEFT BY 1)
2836 ; 0350 .. TO GET THE CORRECT ANSWER
2836 ; 0351 ..
2836 ; 0352 ..MA & AC (AND MQ IF USED) ARE LEFT INCREMENTED BY 2
2836 ; 0353 ..
2836 ; 0354 ..NOTE: P MUST BE ZAP(R0)
2836 ; 0355 ..#####
2836 ; 0356
2836 ; 0357
2836 ; 0358 CSMULT: ..COSINE MULTIPLY
2836 ; 0359 ..DISABLE INTERRUPTS
2836 E07100; 0360 SEX ZAP; DIS ,#00
2839 ; 0361
2839 ; 0362 ..TURN ON MULTIPLIER
2839 61106200; 0363 OUT 1,#10; OUT 2,#00
283D ; 0364
283D ; 0365 ..OUTPUT COSINE VALUE
283D F8DEAC; 0366 LDI A.0(CSVLU); PLO ZAP1
2840 EC6665; 0367 SEX ZAP1; OUT 6; OUT 5
2843 3054; 0368 BR OUTMA ..THEN GOTO CONTINUE
2845 ; 0369
2845 ; 0370 ..-----
2845 ; 0371 EXITM: ..EXIT FFT MULTIPLY
2845 ; 0372 ..TURN OFF MULTIPLIER
2845 E0; 0373 SEX ZAP ..OUT IMMEDIATE
2846 6100; 0374 OUT 1,#00 ..LIGHTS OUT
2848 ; 0375
2848 ; 0376 ..ENABLE INTERRUPTS AND RETURN
2848 7003; 0377 RET ,#03 ..IE=1, X=0, P=3
284A ; 0378
284A ; 0379 ..-----

```

ASP FFT Program  
Program Code

284A ;	0380 SNMULT:	..SINE MULTIPLY
284A ;	0381 ..DISABLE INTERRUPTS	
284A E07100;	0382 SEX ZAP; DIS ,#00	
284D ;	0383	
284D ;	0384 ..TURN ON MULTIPLIER	
284D 6110;	0385 OUT 1,#10	..LATCH SEL4
284F 6200;	0386 OUT 2,#00	..RESET MULTIPLIER
2851 ;	0387	
2851 ;	0388 ..GET OPERAND B FROM M(R(MQ))	
2851 EE;	0389 SEX MQ	
2852 6665;	0390 OUT 6; OUT 5	..OUT HIGH/LOW
2854 ;	0391	
2854 ;	0392 OUTMA:	..OUTPUT M(R(MA))
2854 ;	0393 ..GET OPERAND A	
2854 ED;	0394 SEX MA	
2855 6463;	0395 OUT 4; OUT 3	
2857 ;	0396	
2857 ;	0397 ..COMMAND TO EXECUTE	
2857 E06700;	0398 SEX ZAP; OUT 7,#00	
285A ;	0399	
285A ;	0400 ..NOW RETURN TO MAIN PROGRAM WHILE HARDWARE WORKS	
285A D3;	0401 SEP PC	
285B ;	0402	
285B ;	0403 ..-----	
285B ;	0404 ..WHEN YOU COME BACK, STORE THE RESULT * 2	
285B EF1F;	0405 MLTMDL: SEX AC; INC AC	..GET TO RESULT LOW
285D 6BFE;	0406 INP 3; SHL	..LOW * 2
285F AB73;	0407 PLO ZAP3; STXD	..STORE SHIFTED VALUE
2861 6C7E;	0408 INP 4; SHLC	..HIGH * 2
2863 BB5F;	0409 PHI ZAP3; STR AC	..STORE SHIFTED VALUE
2865 1F1F;	0410 INC AC; INC AC	..GET TO NEXT WORD.1
2867 ;	0411	
2867 ;	0412 ..AND RETURN THROUGH TOP	
2867 3045;	0413 BR EXITM	
2869 ;	0414	
2869 ;	0415	
2869 ;	0416	

ASP FFT Program  
Program Code

```

2869 ; 0417 ..#####
2869 ; 0418 ..ADDSTF.SR ADD & STUFF
2869 ; 0419 ..
2869 ; 0420 ..THIS PROGRAM COMPUTES LOREAL & LOIMAG
2869 ; 0421 .. OR HIREAL & HIIMAG VIA:
2869 ; 0422 ..
2869 ; 0423 ..M(R(ZAP1)) <- R(MP) + M(R(FRP))
2869 ; 0424 ..M(R(ZAP1)+2) <- R(MP) + M(R(FRP)) + 2
2869 ; 0425 ..
2869 ; 0426 ..R(ZAP2) IS USED AS A TEMPORARY STORAGE LOCATION
2869 ; 0427 ..R(MP) IS RETURNED UPDATED BY M(R(FRP))
2869 ; 0428 ..#####
2869 ; 0429
2869 ; 0430
2869 ; 0431 ..RETURN PORTION
2869 ; 0432 ADSTEX: ..ADD & STUFF EXIT
2869 D3; 0433 SEP PC ..RESET TO PC
286A ; 0434
286A ; 0435 ..ENTER HERE
286A ; 0436 ADSTF: ..ADD & STUFF
286A ; 0437 ..ADD AS SHOWN IN FIRST LINE ABOVE
286A E8; 0438 SEX FRP
286B 89; 0439 GLO MP
286C F4; 0440 ADD
286D A9; 0441 PLO MP
286E AA; 0442 PLO ZAP2
286F ; 0443
286F 28; 0444 DEC FRP
2870 99; 0445 GHI MP
2871 74; 0446 ADC
2872 B9; 0447 PHI MP
2873 BA; 0448 PHI ZAP2
2874 ; 0449
2874 ; 0450 ..STORE TO --REAL
2874 5C; 0451 STR ZAP1 ..HIGH BYTE STILL IN D
2875 ; 0452
2875 1C; 0453 INC ZAP1 ..STORE LOW BYTE
2876 8A; 0454 GLO ZAP2
2877 5C; 0455 STR ZAP1
2878 ; 0456
2878 ; 0457 ..INCREMENT VALUE BY 2
2878 1A; 0458 INC ZAP2
2879 1A; 0459 INC ZAP2
287A ; 0460

```



ASP FFT Program  
Program Code

287A ;	0461	..STORE TO --IMAG
287A 1C;	0462	INC ZAP1
287B 9A;	0463	GHI ZAP2
287C 5C;	0464	STR ZAP1
287D ;	0465	
287D 1C;	0466	INC ZAP1
287E 8A;	0467	GLO ZAP2
287F 5C;	0468	STR ZAP1
2880 ;	0469	
2880 ;	0470	..GET R(ZAP1) TO THE NEXT LOCATION
2880 1C;	0471	INC ZAP1
2881 ;	0472	
2881 ;	0473	..THEN RETURN THRU TOP
2881 3069;	0474	BR ADSTEX
2883 ;	0475	
2883 ;	0476	
2883 ;	0477	

ASP FFT Program  
Program Code

```

2883 ; 0478 ..#####
2883 ; 0479 ..NEWVAL.SR NEW VALUE
2883 ; 0480 ..
2883 ; 0481 ..THIS PROGRAM COMPUTES THE NEW COLUMN'S DATA
2883 ; 0482 .. AT POINTERS
2883 ; 0483 ..LOREAL & HIREAL OR LOIMAG & HIIMAG VIA:
2883 ; 0484 ..
2883 ; 0485 ..M(M(R(FRP))) <- M(M(R(FRP))) + M(R(ZAP1))
2883 ; 0486 ..M(M(R(FRP))+4) <- M(M(R(FRP))) - M(R(ZAP1))
2883 ; 0487 ..WHERE:
2883 ; 0488 .. R(FRP) IS THE LO---- POINTER LOCATION
2883 ; 0489 .. M(R(FRP)) IS THE ADDRESS POINTED TO
2883 ; 0490 ..& M(M(R(FRP))) IS THE VALUE AT THAT ADDRESS
2883 ; 0491 ..
2883 ; 0492 ..NOTE: THE REGISTERS WILL BE WRECKED ON RETURN
2883 ; 0493 ..#####
2883 ; 0494
2883 ; 0495
2883 ; 0496
2883 ; 0497 ..RETURN PORTION
2883 ; 0498 EXNWVL: ..EXIT NEW VALUE
2883 D3; 0499 SEP PC
2884 ; 0500
2884 ; 0501 ..ENTER HERE
2884 ; 0502 NEWVAL: ..NEW VALUE
2884 ; 0503
2884 ; 0504 ..LOAD THE VALUE AT R(ZAP1) INTO R(MA)
2884 EC; 0505 SEX ZAP1
2885 72; 0506 LDXA
2886 BD; 0507 PHI MA
2887 72; 0508 LDXA
2888 AD; 0509 PLO MA
2889 ; 0510
2889 ; 0511 ..R(FRP) IS AT THE LO---- POINTER
2889 ; 0512 ..GET R(MP) TO THE ADDRESS R(FRP) POINTS TO
2889 ; 0513 ..THAT'S THE LO---- VALUE ADDRESS
2889 E8; 0514 SEX FRP
288A 72; 0515 LDXA
288B B9; 0516 PHI MP
288C 72; 0517 LDXA
288D A9; 0518 PLO MP
288E ; 0519
288E ; 0520 ..GET TO IT'S LOW BYTE
288E 19; 0521 INC MP
288F ; 0522

```

288F ;	0523 ..DO THE SUBTRACTION
288F ;	0524 ..AND SAVE THE RESULT IN R(AC)
288F E9;	0525 SEX MP
2890 8D;	0526 GLO MA
2891 F5;	0527 SD
2892 AF;	0528 PLO AC
2893 29;	0529 DEC MP
2894 9D;	0530 GHI MA
2895 75;	0531 SDB
2896 BF;	0532 PHI AC
2897 ;	0533
2897 ;	0534 ..THEN DO THE ADD & STORE
2897 19;	0535 INC MP
2898 8D;	0536 GLO MA
2899 F4;	0537 ADD
289A 73;	0538 STXD
289B 9D;	0539 GHI MA
289C 74;	0540 ADC
289D 73;	0541 STXD
289E ;	0542
289E ;	0543 ..GET R(FRP) TO THE HI---- POINTER ADDRESS
289E 18;	0544 INC FRP
289F 18;	0545 INC FRP
28A0 ;	0546
28A0 ;	0547 ..GET R(MP) TO THE ADDRESS R(FRP) POINTS TO
28A0 ;	0548 ..THAT'S THE HI---- VALUE ADDRESS
28A0 E8;	0549 SEX FRP
28A1 72;	0550 LDXA
28A2 B9;	0551 PHI MP
28A3 72;	0552 LDXA
28A4 A9;	0553 PLO MP
28A5 ;	0554
28A5 ;	0555 ..GET TO IT'S LOW BYTE
28A5 19;	0556 INC MP
28A6 ;	0557
28A6 ;	0558 ..STORE THE SUBTRACTION RESULT
28A6 E9;	0559 SEX MP
28A7 8F;	0560 GLO AC
28A8 73;	0561 STXD
28A9 9F;	0562 GHI AC
28AA 73;	0563 STXD
28AB ;	0564
28AB ;	0565 ..THEN RETURN THRU TOP
28AB 3083;	0566 BR EXNWVL
28AD ;	0567
28AD ;	0568
28AD ;	0569

ASP FFT Program  
Program Code

```

28AD ;          0570 ..#####
28AD ;          0571 ..TRSET.SR          TRIG SET
28AD ;          0572 ..
28AD ;          0573 ..THIS ROUTINE COMPUTES THE ADDRESSES OF THE
28AD ;          0574 ..POINTERS FOR THE 256 ENTRY TRIG TABLE
28AD ;          0575 ..THAT IS:
28AD ;          0576 ..
28AD ;          0577 ..TRGPTR <- BITREV (TRGCTR)
28AD ;          0578 ..SNPTR <- TRGFWA + TRGPTR
28AD ;          0579 ..CSPTR <- TRGLWA - TRGPTR
28AD ;          0580 ..
28AD ;          0581 ..IF SNPTR IS PAST TRGLWA THEN IT RETURNS IT BACK
28AD ;          0582 ..      AS MANY AS IT WAS OFF THE END
28AD ;          0583 ..FOR CSPTR IT STORES THE VALUE AT CSVLU
28AD ;          0584 ..IF CSPTR IS BEFORE TRGFWA IT RETURNS IT BACK
28AD ;          0585 ..      AS MANY AS IT WAS IN FRONT OF THE START
28AD ;          0586 ..      AND STORES THAT VALUE, NEGATED, INTO CSVLU
28AD ;          0587 ..
28AD ;          0588 ..THERE IS A SECOND ENTRY POINT USED BY THE SHAPE
28AD ;          0589 ..      ROUTINE, THAT IS AT SHPSET, WHICH USES
28AD ;          0590 ..      THE CSVLU COMPUTING ALGORITHM
28AD ;          0591 ..
28AD ;          0592 ..IT USES R(ZAP1), R(ZAP2), R(ZAP3), R(MQ) & R(FRP)
28AD ;          0593 ..IT ASSUMES R(MP) HAS TRGCTR VALUE
28AD ;          0594 ..#####
28AD ;          0595
28AD ;          0596
28AD ;          0597 ..LOCAL VARIABLES
28AD ;          0598
28AD ;          0599          TRGCTR=CSVLU+12 ..TRIG COUNTER
28AD ;          0600          CSPTR=TRGCTR+2 ..COSINE POINTER
28AD ;          0601          SNPTR=CSPTR+2 ..SINE POINTER
28AD ;          0602          TRGPTR=SNPTR+2 ..TRIG POINTER
28AD ;          0603
28AD ;          0604
28AD ;          0605 ..RETURN PORTION
28AD ;          0606 EXTRST:          ..EXIT TRIG SET
28AD D3;          0607          SEP PC          ..RESET TO PC
28AE ;          0608
28AE ;          0609 ..ENTER HERE
28AE ;          0610 TRGSET:          ..TRIG SET
28AE ;          0611
28AE ;          0612 ..USE R(ZAP3) FOR TRGPTR REGISTER
28AE ;          0613 ..CLEAR IT
28AE F800;          0614          LDI #00
28B0 BB;          0615          PHI ZAP3
28B1 AB;          0616          PLO ZAP3
28B2 ;          0617

```



ASP FFT Program  
Program Code

28B2 ;	0618	..LOAD SHIFT VALUE ( 10 ) INTO R(ZAP2)	
28B2 F800;	0619	LDI #00	
28B4 BA;	0620	PHI ZAP2	
28B5 F80A;	0621	LDI #0A	
28B7 AA;	0622	PLO ZAP2	
28B8 ;	0623		
28B8 ;	0624	..SHIFT TRGCTR INTO TRGPTR THRU DF	
28B8 ;	0625	SHFARO:	..SHIFT AROUND
28B8 99;	0626	GHI MP	..SHIFT RIGHT BIT OF TRGCTR
28B9 F6;	0627	SHR	
28BA B9;	0628	PHI MP	
28BB 89;	0629	GLO MP	
28BC 76;	0630	SHRC	..INTO DF
28BD A9;	0631	PLO MP	
28BE ;	0632		
28BE 8B;	0633	GLO ZAP3	..SHIFT SAVED BIT LEFT
28BF 7E;	0634	SHLC	..INTO TRGPTR
28C0 AB;	0635	PLO ZAP3	
28C1 9B;	0636	GHI ZAP3	
28C2 7E;	0637	SHLC	
28C3 BB;	0638	PHI ZAP3	
28C4 ;	0639		
28C4 ;	0640	..COUNT DOWN VIA R(ZAP2) 10 TIMES	
28C4 2A;	0641	DEC ZAP2	
28C5 8A;	0642	GLO ZAP2	..IF NOT 0
28C6 3AB8;	0643	BNZ SHFARO	..SHIFT AROUND AGAIN
28C8 ;	0644		
28C8 ;	0645	..ELSE, STORE TRGPTR VALUE	
28C8 ;	0646	..GET R(FRP) TO TRGPTR	
28C8 F8F1;	0647	LDI A.0(TRGPTR+1)	
28CA A8;	0648	PLO FRP	
28CB ;	0649		
28CB ;	0650	..AND STORE	
28CB E8;	0651	SEX FRP	
28CC 8B;	0652	GLO ZAP3	
28CD 73;	0653	STXD	
28CE 9B;	0654	GHI ZAP3	
28CF 73;	0655	STXD	
28D0 ;	0656	..LEAVE R(FRP) AT SNPTR	
28D0 ;	0657		
28D0 ;	0658	..NOW DO SNPTR <- TRGFWA + TRGPTR	
28D0 8BFCFE73;	0659	GLO ZAP3; ADI A.0(TRGFWA); STXD	
28D4 9B7C2658;	0660	GHI ZAP3; ADCI A.1(TRGFWA); STR FRP	
28D8 ;	0661		

ASP FFT Program  
Program Code

28D8 ;	0662 ..CHECK FOR SNPTR > TRGLWA
28D8 18;	0663 INC FRP
28D9 F8FEF7AA;	0664 LDI A.0(TRGLWA); SM; PLO ZAP2
28DD 28;	0665 DEC FRP
28DE F82777BA;	0666 LDI A.1(TRGLWA); SMB; PHI ZAP2
28E2 C328EE;	0667 LBDF CSCMPT ..IF SNPTR <= TRGLWA, THEN
28E5 ;	0668 ..GOTO COSINE COMPUTE
28E5 ;	0669
28E5 ;	0670 ..ELSE ADD NEGATIVE OFFSET TO TRGLWA, THEN STORE
28E5 18;	0671 INC FRP
28E6 8AFCFE73;	0672 GLO ZAP2; ADI A.0(TRGLWA); STXD
28EA 9A7C2758;	0673 GHI ZAP2; ADCI A.1(TRGLWA); STR FRP
28EE ;	0674
28EE ;	0675 CSCMPT: ..COSINE COMPUTE
28EE ;	0676 ..FIRST DO CSPTR <- TRGLWA - TRGPTR
28EE 28;	0677 DEC FRP
28EF 8BFDFEAE;	0678 GLO ZAP3; SDI A.0(TRGLWA); PLO MQ
28F3 9B7D27BE;	0679 GHI ZAP3; SDBI A.1(TRGLWA); PHI MQ
28F7 ;	0680
28F7 ;	0681 SHPSET: ..SHAPE SET
28F7 ;	0682 ..ENTRY POINT FOR SHAPE SET
28F7 ;	0683 ..COSINE POINTER VALUE IS IN R(MQ)
28F7 ;	0684 ..R(FRP) IS AT COSINE POINTER (LOW)
28F7 ;	0685
28F7 ;	0686 ..STORE POSSIBLE COSINE POINTER ADDRESS
28F7 E8;	0687 SEX FRP
28F8 8EAB73;	0688 GLO MQ; PLO ZAP3; STXD
28FB 9EBB58;	0689 GHI MQ; PHI ZAP3; STR FRP
28FE ;	0690
28FE ;	0691 ..SET FLAG FOR POSITIVE STORE
28FE F800AC;	0692 LDI #00; PLO ZAP1
2901 ;	0693
2901 ;	0694 ..CHECK FOR CSPTR < TRGFWA
2901 8EFFFFEAA;	0695 GLO MQ; SMI A.0(TRGFWA); PLO ZAP2
2905 9E7F26BA;	0696 GHI MQ; SMBI A.1(TRGFWA); PHI ZAP2
2909 C3291A;	0697 LBDF CSVLST ..IF PTR => FWA, THEN
290C ;	0698 ..GOTO COSINE VALUE STORE
290C ;	0699
290C ;	0700 ..ELSE "SUBTRACT NEGATIVE" (ADD) OFFSET TO TRGFWA
290C ;	0701 ..THEN STORE CORRECTED ADDRESS
290C 18;	0702 INC FRP
290D 8AFDFE;	0703 GLO ZAP2; SDI A.0(TRGFWA)
2910 AB73;	0704 PLO ZAP3; STXD
2912 9A7D26;	0705 GHI ZAP2; SDBI A.1(TRGFWA)
2915 BB58;	0706 PHI ZAP3; STR FRP
2917 ;	0707
2917 ;	0708 ..SET FLAG FOR NEGATIVE STORE
2917 F8FFAC;	0709 LDI #FF; PLO ZAP1
291A ;	0710

ASP FFT Program  
Program Code

291A ;	0711 CSVLST:	..COSINE VALUE STORE
291A ;	0712 ..GET COSINE VALUE AT ADDRESS FOR STORING	
291A 4BBA4BAA;	0713 LDA ZAP3; PHI ZAP2; LDA ZAP3; PLO ZAP2	
291E ;	0714	
291E ;	0715 ..GET R(FRP) TO "COSINE VALUE" LOCATION	
291E F8DFA8;	0716 LDI A.0(CSVLU+1); PLO FRP	
2921 ;	0717	
2921 ;	0718 ..CHECK FOR NEGATE	
2921 8CCA292C;	0719 GLO ZAP1; LBNZ FLPSTR ..IF NEGATE TRUE, THEN	
2925 ;	0720 ..GOTO FLIP & STORE	
2925 ;	0721	
2925 ;	0722 ..ELSE STORE AS IS	
2925 8A739A58;	0723 GLO ZAP2; STXD; GHI ZAP2; STR FRP	
2929 C028AD;	0724 LBR EXTRST ..AND RETURN THROUGH TOP	
292C ;	0725	
292C ;	0726 FLPSTR:	..FLIP AND STORE
292C ;	0727 ..NEGATE, THEN STORE	
292C 8AFD0073;	0728 GLO ZAP2; SDI #00; STXD	
2930 9A7D0058;	0729 GHI ZAP2; SDBI #00; STR FRP	
2934 C028AD;	0730 LBR EXTRST ..AND RETURN THROUGH TOP	
2937 ;	0731	
2937 ;	0732	
2937 ;	0733	
2937 ;	0734 ..#####	
2937 ;	0735 ..END OF SUBROUTINES.	

ASP FFT Program  
Program Code

A.2.3.2 DTASCL (Data Scale)

```

0000 ;          0001
0000 ;          0002 ..DTASCL.SR          26 MAY 80          5:40 PM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..DTASCL.SR          DATA SCALE
0000 ;          0244 ..
0000 ;          0245 ..THIS ROUTINE SCALES THE DATA SO THAT THE GREATEST
0000 ;          0246 ..MAGNITUDE IS BETWEEN +MAXVALUE & -MAXVALUE
0000 ;          0247 ..ONLY THE TOP BYTE IS CHECKED AND IF
0000 ;          0248 ..THIS MAXSHIFT = 7, MAXVALUE = #7FFF
0000 ;          0249 ..          6,          #3FFF,
0000 ;          0250 ..          5,          #1FFF, ETC.
0000 ;          0251 ..
0000 ;          0252 ..FIRST THE FIFO IS SCANNED FROM START FOR LENGTH
0000 ;          0253 ..VALUES TO FIND THE HIGHEST SIGNIFICANT PLACE,
0000 ;          0254 ..OR UNTIL A NUMBER WHERE MSB IS ACTIVE
0000 ;          0255 ..          (FXXX-CXXX OR 4XXX-7XXX)
0000 ;          0256 ..THEN EVERY WORD IS SHIFTED (IF REQUIRED)
0000 ;          0257 ..
0000 ;          0258 ..IT IS ASSUMED THAT START POINTER IS IN R(MP)
0000 ;          0259 ..LENGTH POINTER IS AT M(R(FRP))
0000 ;          0260 ..THIS MAXIMUM SHIFT VALUE IS IN R(ZAP2)
0000 ;          0261 ..BOUNDARY VALUE POINTER IS IN R(ZAP1)
0000 ;          0262 ..AND START BOUNDARY ADDRESS AT M(R(ZAP1)+2)
0000 ;          0263 ..
0000 ;          0264 ..NO REGISTERS ARE SAVED FOR RETURN
0000 ;          0265 ..#####
0000 ;          0266
0000 ;          0267
0000 ;          0268 ..INTERNAL VARIABLES
0000 ;          0269 ..SCALE VALUES
0000 ;          0270
0000 ;          0271 THBDFL=YSLWA+1 ..THIS BOUNDARY FLAG
0000 ;          0272 BDOKRM=THBDFL+2 ..BOUNDARY OK REMEMBER
0000 ;          0273 LNFREE=BDOKRM+2 ..LENGTH FREE
0000 ;          0274 UNDRSH=LNFREE+2 ..UNDER SHIFT VALUE
0000 ;          0275 SCSTRT=UNDRSH+2 ..SCALE START ADDR
0000 ;          0276 SCLN=SCSTRT+2 ..SCALE LENGTH COUNT
0000 ;          0277
0000 ;          0278

```



0000 ;	0279 ..*****
0000 ;	0280 ..BEGIN HERE
0000 ;	0281 ..VECTORS TO INTERNAL ROUTINES
0000 ;	0282 ORG #2960
2960 C02967;	0283 LBR DTASCL ..GOTO DATA SCALE
2963 ;	0284
2963 ;	0285 ..RETURN PORTION
2963 ;	0286 EXDTSC: ..EXIT DATA SCALE
2963 ;	0287
2963 ;	0288 ..RETURN R(ZAP1) TO FRAM
2963 F830BC;	0289 LDI A.1(FRAM); PHI ZAP1
2966 ;	0290
2966 ;	0291 ..THEN RETURN TO CALLING PROGRAM
2966 D3;	0292 SEP PC ..RESET TO PC
2967 ;	0293
2967 ;	0294
2967 ;	0295 ..ENTER HERE
2967 ;	0296
2967 ;	0297 DTASCL: ..DATA SCALE
2967 ;	0298 ..GET START ADDRESS INTO R(MA)
2967 E9;	0299 SEX MP
2968 72;	0300 LDXA
2969 BD;	0301 PHI MA
296A F0;	0302 LDX
296B AD;	0303 PLO MA
296C ;	0304
296C ;	0305 ..GET LENGTH VALUE INTO R(MP)
296C E8;	0306 SEX FRP
296D 72;	0307 LDXA
296E B9;	0308 PHI MP
296F F0;	0309 LDX
2970 A9;	0310 PLO MP
2971 ;	0311
2971 ;	0312 ..PASS BOUNDARY VALUE POINTER
2971 ;	0313 ..FROM R(ZAP1) TO R(FRP)
2971 9CB8;	0314 GHI ZAP1; PHI FRP
2973 8CA8;	0315 GLO ZAP1; PLO FRP
2975 ;	0316
2975 ;	0317 ..SAVE THESE VALUES FOR USE DURING SCALING
2975 ;	0318 ..GET R(ZAP1) TO SCALE LENGTH
2975 F83E;	0319 LDI A.1(SCLN+1)
2977 BC;	0320 PHI ZAP1
2978 F80C;	0321 LDI A.0(SCLN+1)
297A AC;	0322 PLO ZAP1
297B ;	0323
297B ;	0324 ..SAVE LENGTH COUNT
297B EC;	0325 SEX ZAP1
297C 89;	0326 GLO MP
297D 73;	0327 STXD
297E 99;	0328 GHI MP
297F 73;	0329 STXD

ASP FFT Program  
Program Code

2980 ;	0330	
2980 ;	0331	..SAVE START ADDRESS
2980 8D;	0332	GLO MA
2981 73;	0333	STXD
2982 9D;	0334	GHI MA
2983 73;	0335	STXD
2984 ;	0336	
2984 ;	0337	..SAVE UNDERSHIFT VALUE
2984 ;	0338	..(ABSOLUTE MAX SHIFT, #07 - THIS MAX SHIFT)
2984 8A;	0339	GLO ZAP2
2985 FD07;	0340	SDI #07
2987 73;	0341	STXD
2988 9A;	0342	GHI ZAP2
2989 7D00;	0343	SDBI #00
298B 73;	0344	STXD
298C ;	0345	
298C ;	0346	..GET LENGTH FREE STORED & INTO R(ZAP2)
298C ;	0347	..(LENGTH FREE = BOUNDARY - START ADDRESS)
298C E8;	0348	SEX FRP
298D 18;	0349	INC FRP
298E 8D;	0350	GLO MA
298F F5;	0351	SD
2990 5C;	0352	STR ZAP1
2991 2C;	0353	DEC ZAP1
2992 AA;	0354	PLO ZAP2
2993 28;	0355	DEC FRP
2994 9D;	0356	GHI MA
2995 75;	0357	SDB
2996 5C;	0358	STR ZAP1
2997 BA;	0359	PHI ZAP2
2998 ;	0360	
2998 ;	0361	..AND LEAVE R(FRP) AT FWA
2998 18;	0362	INC FRP
2999 18;	0363	INC FRP
299A ;	0364	
299A ;	0365	..CHECK FOR BOUNDARY OK
299A ;	0366	..(IF SCALE LENGTH <= LENGTH FREE)
299A EC;	0367	SEX ZAP1
299B 1C;	0368	INC ZAP1
299C 89;	0369	GLO MP
299D F5;	0370	SD
299E 2C;	0371	DEC ZAP1
299F 99;	0372	GHI MP
29A0 75;	0373	SDB
29A1 CB29B1;	0374	LBNF BDRYBD ..IF SCALE <= FREE, THEN
29A4 ;	0375	..GOTO BOUNDARY BAD
29A4 ;	0376	

29A4 ;	0377	..BOUNDARY GOOD
29A4 ;	0378	..STORE TRUE (#FF) INTO BOUNDARY FLAG REMEMBER
29A4 2C;	0379	DEC ZAP1
29A5 F8FF;	0380	LDI #FF
29A7 73;	0381	STXD
29A8 ;	0382	
29A8 ;	0383	..GET R(ZAP1) TO BOUNDARY OK FLAG
29A8 F801;	0384	LDI A.0(THBDFL)
29AA AC;	0385	PLO ZAP1
29AB ;	0386	
29AB ;	0387	..STORE TRUE (#FF) THERE
29AB F8FF;	0388	LDI #FF
29AD 5C;	0389	STR ZAP1
29AE ;	0390	
29AE ;	0391	..AND JUMP TO GET SET UP
29AE C029BB;	0392	LBR GTSTUP
29B1 ;	0393	
29B1 ;	0394	BDYBDB: ..BOUNDARY BAD
29B1 ;	0395	..STORE FALSE (#00) INTO BOUNDARY FLAG REMEMBER
29B1 2C;	0396	DEC ZAP1
29B2 F800;	0397	LDI #00
29B4 73;	0398	STXD
29B5 ;	0399	
29B5 ;	0400	..GET R(ZAP1) TO BOUNDARY OK FLAG
29B5 F801;	0401	LDI A.0(THBDFL)
29B7 AC;	0402	PLO ZAP1
29B8 ;	0403	
29B8 ;	0404	..STORE FALSE (#00) THERE
29B8 F800;	0405	LDI #00
29BA 5C;	0406	STR ZAP1
29BB ;	0407	
29BB ;	0408	..AND FALL THROUGH TO GET SET UP
29BB ;	0409	
29BB ;	0410	GTSTUP: ..GET SET UP
29BB ;	0411	..GET ABSOLUTE MAXIMUM SHIFT COUNT INTO R(MQ)
29BB F800;	0412	LDI #00
29BD BE;	0413	PHI MQ
29BE F807;	0414	LDI #07
29C0 AE;	0415	PLO MQ
29C1 ;	0416	
29C1 ;	0417	..GET #FF INTO R(AND) MASK, (HIGH R(AC))
29C1 F8FF;	0418	LDI #FF
29C3 BF;	0419	PHI AC
29C4 ;	0420	
29C4 ;	0421	..GET #00 INTO R(OR) MASK, (LOW R(AC))
29C4 F800;	0422	LDI #00
29C6 AF;	0423	PLO AC
29C7 ;	0424	
29C7 ;	0425	

ASP FFT Program  
Program Code

29C7 ;	0426	..*****	
29C7 ;	0427		
29C7 ;	0428	MEMTST:	..MEMORY TEST
29C7 ;	0429	..SEE IF THE R(AND/OR) MASK ALLOWS THIS MANY SHIFTS	
29C7 ;	0430	..EITHER TRUE - NEXT SHIFT	
29C7 ;	0431	.. OR FALSE - SHIFT LESS	
29C7 ;	0432		
29C7 ;	0433	..POINT X TO MEMORY POINTER, R(MA)	
29C7 ED;	0434	SEX MA	
29C8 ;	0435		
29C8 ;	0436	..ASSUME MEMORY VALUE POSITIVE	
29C8 ;	0437	..MEM.AND.R(AND)	
29C8 9F;	0438	GHI AC	
29C9 F2;	0439	AND	
29CA ;	0440		
29CA ;	0441	..IF #00, THIS SHIFT IS OK	
29CA ;	0442	..THEN GOTO NEXT MEMORY CHECK	
29CA C229D8;	0443	LBZ NXTCHK	
29CD ;	0444		
29CD ;	0445	..ELSE, CHECK SIGN BIT	
29CD FE;	0446	SHL	
29CE ;	0447		
29CE ;	0448	..IF POSITIVE AND NOT #00	
29CE ;	0449	..THEN YOU MUST SHIFT LESS	
29CE CB29FE;	0450	LBNF SHFTLS	
29D1 ;	0451		
29D1 ;	0452	..ELSE, MEMORY IS NEGATIVE	
29D1 ;	0453	..MEM.OR.R(OR)	
29D1 8F;	0454	GLO AC	
29D2 F1;	0455	OR	
29D3 ;	0456		
29D3 ;	0457	..IF NOT #FF, YOU MUST SHIFT LESS	
29D3 FBFF;	0458	XRI #FF	
29D5 CA29FE;	0459	LBNZ SHFTLS	
29D8 ;	0460		
29D8 ;	0461	..ELSE, FALL THROUGH TO TEST FOR LAST	
29D8 ;	0462		
29D8 ;	0463	..DECREMENT SCALE LENGTH COUNTER, CHECK FOR #00	
29D8 ;	0464	NXTCHK:	..NEXT MEMORY CHECK
29D8 29;	0465	DEC MP	
29D9 29;	0466	DEC MP	
29DA 89;	0467	GLO MP	
29DB CA29E2;	0468	LBNZ NXTMEM	..NO, GOTO NEXT MEMORY
29DE 99;	0469	GHI MP	
29DF C22A0C;	0470	LBZ NOWSCL	..DONE, SO GOTO NOW SCALE
29E2 ;	0471		
29E2 ;	0472	..NEXT MEMORY, GET R(MA) TO NEXT HIBYTE	
29E2 1D;	0473	NXTMEM: INC MA	
29E3 1D;	0474	INC MA	
29E4 ;	0475		



29E4 ;	0476	..CHECK BOUNDARY OK FLAG
29E4 0C;	0477	LDN ZAP1
29E5 CA29C7;	0478	LBNZ MEMTST ..IF TRUE,
29E8 ;	0479	..JUMP BACK TO MEMORY TEST
29E8 ;	0480	
29E8 ;	0481	..ELSE CHECK AGAINST BOUNDARY
29E8 ;	0482	..DECREMENT FREE LENGTH COUNTER
29E8 2A;	0483	DEC ZAP2
29E9 2A;	0484	DEC ZAP2
29EA ;	0485	
29EA ;	0486	..IF NOT ZERO, JUMP BACK TO MEMORY TEST
29EA 8A;	0487	GLO ZAP2
29EB CA29C7;	0488	LBNZ MEMTST
29EE 9A;	0489	GHI ZAP2
29EF CA29C7;	0490	LBNZ MEMTST
29F2 ;	0491	
29F2 ;	0492	..ELSE, SET R(MA) TO FIFO FWA
29F2 E8;	0493	SEX FRP
29F3 72;	0494	LDXA
29F4 BD;	0495	PHI MA
29F5 F0;	0496	LDX
29F6 AD;	0497	PLO MA
29F7 28;	0498	DEC FRP
29F8 ;	0499	..AND RETURN R(FRP) TO FWA
29F8 ;	0500	
29F8 ;	0501	..SET BOUNDARY OK FLAG TRUE
29F8 F8FF;	0502	LDI #FF
29FA 5C;	0503	STR ZAP1
29FB ;	0504	
29FB ;	0505	..AND JUMP BACK TO MEMORY TEST
29FB C029C7;	0506	LBR MEMTST
29FE ;	0507	
29FE ;	0508	SHFTLS: ..SHIFT LESS
29FE ;	0509	..COUNT DOWN ABSOLUTE MAXSHIFT BY 1
29FE 2E;	0510	DEC MQ
29FF ;	0511	
29FF ;	0512	..IF DOWN TO THE LIMIT (#00)
29FF 8E;	0513	GLO MQ
2A00 ;	0514	..GOTO NOW SCALE
2A00 C22A0C;	0515	LBZ NOWSCL
2A03 ;	0516	
2A03 ;	0517	..ELSE, SHIFT A ZERO LEFT INTO R(AND)
2A03 ;	0518	..AND LEAVE A 1 IN DF
2A03 9F;	0519	GHI AC
2A04 FE;	0520	SHL
2A05 BF;	0521	PHI AC
2A06 ;	0522	
2A06 ;	0523	..SHIFT THAT 1 LEFT INTO R(OR)
2A06 8F;	0524	GLO AC
2A07 7E;	0525	SHLC
2A08 AF;	0526	PLO AC

ASP FFT Program  
Program Code

2A09 ;	0527	
2A09 ;	0528	..THEN GO BACK TO TEST AGAIN
2A09 C029C7;	0529	LBR MEMTST
2A0C ;	0530	
2A0C ;	0531	
2A0C ;	0532	..*****
2A0C ;	0533	
2A0C ;	0534	NOWSCL: ..NOW SCALE
2A0C ;	0535	..GET R(ZAP1) TO UNDERSHIFT
2A0C ;	0536	..(ABSOLUTE MAX SHIFT - THIS MAX SHIFT)
2A0C F808;	0537	LDI A.0(UNDRSH+1)
2A0E AC;	0538	PLO ZAP1
2A0F ;	0539	
2A0F ;	0540	..SUBTRACT UNDERSHIFT VALUE AT M(R(ZAP1))
2A0F ;	0541	..FROM 'SHIFT TO ABSOLUTE MAX VALUE' IN R(MQ)
2A0F ;	0542	..FOR THIS SHIFT VALUE. PUT INTO R(MQ)
2A0F EC;	0543	SEX ZAP1
2A10 8E;	0544	GLO MQ
2A11 F7;	0545	SM
2A12 AE;	0546	PLO MQ
2A13 2C;	0547	DEC ZAP1
2A14 9E;	0548	GHI MQ
2A15 77;	0549	SMB
2A16 BE;	0550	PHI MQ
2A17 ;	0551	
2A17 ;	0552	..CHECK SHIFT VALUE, IF #00 RETURN
2A17 8E;	0553	GLO MQ
2A18 C22963;	0554	LBZ EXDTSC
2A1B ;	0555	
2A1B ;	0556	..ELSE, SUBTRACT THIS SHIFT FROM TOTAL SHIFT VALUE
2A1B ;	0557	..GET R(ZAP2) TO TOTAL SHIFT LOCATION
2A1B F830BA;	0558	LDI A.1(TOTLSH+1); PHI ZAP2
2A1E F8DBAA;	0559	LDI A.0(TOTLSH+1); PLO ZAP2
2A21 EA;	0560	SEX ZAP2
2A22 ;	0561	
2A22 ;	0562	..SUBTRACT THIS SHIFT
2A22 8E;	0563	GLO MQ
2A23 F5;	0564	SD
2A24 73;	0565	STXD
2A25 9E;	0566	GHI MQ
2A26 75;	0567	SDB
2A27 73;	0568	STXD
2A28 ;	0569	
2A28 ;	0570	..SET UP FOR SHIFTING
2A28 ;	0571	..GET R(ZAP1) TO BOUNDARY OK REMEMBER
2A28 F804;	0572	LDI A.0(BDOKRM+1)
2A2A AC;	0573	PLO ZAP1
2A2B ;	0574	

2A2B ;	0575	..LOAD VALUE AND SAVE IN TEMPORARY R(AC)	
2A2B EC;	0576	SEX ZAP1	
2A2C 72;	0577	LDXA	
2A2D BF;	0578	PHI AC	
2A2E ;	0579		
2A2E ;	0580	..LOAD FREE LENGTH INTO R(ZAP2)	
2A2E 72;	0581	LDXA	
2A2F BA;	0582	PHI ZAP2	
2A30 72;	0583	LDXA	
2A31 AA;	0584	PLO ZAP2	
2A32 ;	0585		
2A32 ;	0586	..LOAD SCALE START INTO R(MA)	
2A32 1C;	0587	INC ZAP1	
2A33 1C;	0588	INC ZAP1	
2A34 72;	0589	LDXA	
2A35 BD;	0590	PHI MA	
2A36 72;	0591	LDXA	
2A37 AD;	0592	PLO MA	
2A38 ;	0593		
2A38 ;	0594	..LOAD SCALE LENGTH INTO R(MP)	
2A38 72;	0595	LDXA	
2A39 B9;	0596	PHI MP	
2A3A 72;	0597	LDXA	
2A3B A9;	0598	PLO MP	
2A3C ;	0599		
2A3C ;	0600	..GET R(ZAP1) TO BOUNDARY OK FLAG	
2A3C F801;	0601	LDI A.0(THBDFL)	
2A3E AC;	0602	PLO ZAP1	
2A3F ;	0603		
2A3F ;	0604	..STORE REMEMBERED FLAG THERE	
2A3F 9F;	0605	GHI AC	
2A40 5C;	0606	STR ZAP1	
2A41 ;	0607		
2A41 ;	0608	..GET THIS SHIFT VALUE BACK	
2A41 8E;	0609	GLO MQ	
2A42 ;	0610		
2A42 ;	0611	..CHECK SIGN BIT	
2A42 FE;	0612	SHL	
2A43 ;	0613		
2A43 ;	0614	..IF NEGATIVE, GOTO SHIFT DOWN	
2A43 C32AB7;	0615	LBDF SHDOWN	
2A46 ;	0616		
2A46 ;	0617	..ELSE POSITIVE, CHECK FOR UP BY 1	
2A46 76;	0618	SHRC	
2A47 FD01;	0619	SDI #01	
2A49 CA2A7A;	0620	LBNZ SHUPMR	..SHIFT UP MORE
2A4C ;	0621		..IF > 1
2A4C ;	0622		
2A4C ;	0623		

ASP FFT Program  
Program Code

2A4C ;	0624 SHUPON:	..SHIFT UP ONE
2A4C ED;	0625	SEX MA
2A4D 72;	0626	LDXA
2A4E BF;	0627	PHI AC
2A4F FO;	0628	LDX
2A50 FE;	0629	SHL
2A51 73;	0630	STXD
2A52 9F;	0631	GHI AC
2A53 7E;	0632	SHLC
2A54 5D;	0633	STR MA
2A55 ;	0634	
2A55 ;	0635	..CHECK FOR DONE
2A55 29;	0636	DEC MP
2A56 29;	0637	DEC MP
2A57 89;	0638	GLO MP
2A58 CA2A5F;	0639	LBNZ NXUPON
2A5B 99;	0640	GHI MP
2A5C C22963;	0641	LBZ EXDTSC
2A5F ;	0642	..IF 0, EXIT DATA SCALE
2A5F ;	0643	..ELSE, GET TO NEXT MEMORY
2A5F ;	0644 NXUPON:	..NEXT MEMORY UP ONE
2A5F 1D;	0645	INC MA
2A60 1D;	0646	INC MA
2A61 ;	0647	
2A61 ;	0648	..CHECK BOUNDARY OK FLAG
2A61 0C;	0649	LDN ZAP1
2A62 CA2A4C;	0650	LBNZ SHUPON
2A65 ;	0651	..IF BOUNDARY OK,
2A65 ;	0652	..GOTO SHIFT AGAIN
2A65 ;	0653	..ELSE, DECREMENT FREE LENGTH COUNTER
2A65 2A;	0654	DEC ZAP2
2A66 2A;	0655	DEC ZAP2
2A67 ;	0656	
2A67 ;	0657	..IF NOT ZERO, GOTO SHIFT AGAIN
2A67 8A;	0658	GLO ZAP2
2A68 CA2A4C;	0659	LBNZ SHUPON
2A6B 9A;	0660	GHI ZAP2
2A6C CA2A4C;	0661	LBNZ SHUPON
2A6F ;	0662	
2A6F ;	0663	..ELSE, SET R(MA) TO FIFO FWA
2A6F E8;	0664	SEX FRP
2A70 72;	0665	LDXA
2A71 BD;	0666	PHI MA
2A72 FO;	0667	LDX
2A73 AD;	0668	PLO MA
2A74 ;	0669	
2A74 ;	0670	..SET BOUNDARY OK FLAG TRUE
2A74 F8FF;	0671	LDI #FF
2A76 5C;	0672	STR ZAP1
2A77 ;	0673	



2A77 ;	0674	..AND SHIFT AGAIN	
2A77 C02A4C;	0675	LBR SHUPON	
2A7A ;	0676		
2A7A ;	0677		
2A7A ;	0678	SHUPMR:	..SHIFT UP MORE THAN ONE
2A7A ;	0679	..MOVE WORKING SHIFT VALUE INTO R(ZAP3)	
2A7A 9E;	0680	GHI MQ	
2A7B BB;	0681	PHI ZAP3	
2A7C 8E;	0682	GLO MQ	
2A7D AB;	0683	PLO ZAP3	
2A7E ;	0684		
2A7E ;	0685	..NOW GET DATA WORD	
2A7E ED;	0686	SEX MA	
2A7F 72;	0687	LDXA	
2A80 BF;	0688	PHI AC	
2A81 F0;	0689	LDX	
2A82 AF;	0690	PLO AC	
2A83 ;	0691		
2A83 ;	0692	..NOW SHIFT IT LEFT	
2A83 8F;	0693	NOWSHL: GLO AC	
2A84 FE;	0694	SHL	
2A85 AF;	0695	PLO AC	
2A86 9F;	0696	GHI AC	
2A87 7E;	0697	SHLC	
2A88 BF;	0698	PHI AC	
2A89 ;	0699		
2A89 ;	0700	..ENOUGH SHIFTS ?	
2A89 2B;	0701	DEC ZAP3	
2A8A 8B;	0702	GLO ZAP3	
2A8B CA2A83;	0703	LBZL NOWSHL	..IF NOT, SHIFT AGAIN
2A8E ;	0704		
2A8E ;	0705	..THEN STORE BACK	
2A8E 8F;	0706	GLO AC	
2A8F 73;	0707	STXD	
2A90 9F;	0708	GHI AC	
2A91 5D;	0709	STR MA	
2A92 ;	0710		
2A92 ;	0711	..CHECK FOR DONE	
2A92 29;	0712	DEC MP	..DECREMENT LENGTH COUNT
2A93 29;	0713	DEC MP	
2A94 89;	0714	GLO MP	
2A95 CA2A9C;	0715	LBZL NXUPMR	..NOT 0, GOTO NEXT UP MORE
2A98 99;	0716	GHI MP	
2A99 C22963;	0717	LBZ EXDTSC	..IF 0, EXIT DATA SCALE
2A9C ;	0718		
2A9C ;	0719	..ELSE, GET TO NEXT MEMORY	
2A9C ;	0720	NXUPMR:	..NEXT MEMORY UP MORE
2A9C 1D;	0721	INC MA	
2A9D 1D;	0722	INC MA	
2A9E ;	0723		

ASP FFT Program  
Program Code

2A9E ;	0724	..CHECK BOUNDARY OK FLAG	
2A9E 0C;	0725	LDN ZAP1	
2A9F CA2A7A;	0726	LBNZ SHUPMR	..IF BOUNDARY OK,
2AA2 ;	0727		..GOTO SHIFT AGAIN
2AA2 ;	0728		
2AA2 ;	0729	..ELSE, DECREMENT FREE LENGTH COUNTER	
2AA2 2A;	0730	DEC ZAP2	
2AA3 2A;	0731	DEC ZAP2	
2AA4 ;	0732		
2AA4 ;	0733	..IF NOT ZERO, GOTO SHIFT AGAIN	
2AA4 8A;	0734	GLO ZAP2	
2AA5 CA2A7A;	0735	LBNZ SHUPMR	
2AA8 9A;	0736	GHI ZAP2	
2AA9 CA2A7A;	0737	LBNZ SHUPMR	
2AAC ;	0738		
2AAC ;	0739	..ELSE, SET R(MA) TO FIFO FWA	
2AAC E8;	0740	SEX FRP	
2AAD 72;	0741	LDXA	
2AAE BD;	0742	PHI MA	
2AAF FO;	0743	LDX	
2AB0 AD;	0744	PLO MA	
2AB1 ;	0745		
2AB1 ;	0746	..SET BOUNDARY OK FLAG TRUE	
2AB1 F8FF;	0747	LDI #FF	
2AB3 5C;	0748	STR ZAP1	
2AB4 ;	0749		
2AB4 ;	0750	..AND SHIFT AGAIN	
2AB4 C02A7A;	0751	LBR SHUPMR	
2AB7 ;	0752		
2AB7 ;	0753		
2AB7 ;	0754	SHDOWN:	..SHIFT DOWN
2AB7 ;	0755	..NEGATIVE SHIFT, CHECK FOR DOWN BY 1	
2AB7 76;	0756	SHRC	
2AB8 FC01;	0757	ADI #01	
2ABA CA2AEF;	0758	LBNZ SHDNMR	..SHIFT DOWN MORE
2ABD ;	0759		..IF < -1
2ABD ;	0760		
2ABD ;	0761	SHDNON:	..SHIFT DOWN BY 1
2ABD ;	0762		..(WITH SIGN EXTEND)
2ABD OD;	0763	LDN MA	
2ABE FE;	0764	SHL	
2ABF CB2AC4;	0765	LBNF DONRSR	
2AC2 F901;	0766	ORI #01	

ASP FFT Program  
Program Code

2AC4 ;	0767	DONRSR:	..DOWN ONE RING SHIFT RIGHT
2AC4 76;	0768	RSHR	
2AC5 76;	0769	RSHR	
2AC6 5D;	0770	STR MA	
2AC7 1D;	0771	INC MA	
2AC8 0D;	0772	LDN MA	
2AC9 76;	0773	SHRC	
2ACA 5D;	0774	STR MA	
2ACB 1D;	0775	INC MA	
2ACC ;	0776		
2ACC ;	0777	..CHECK FOR DONE	
2ACC 29;	0778	DEC MP	..DECREMENT LENGTH COUNT
2ACD 29;	0779	DEC MP	
2ACE 89;	0780	GLO MP	
2ACF CA2AD6;	0781	LBNZ CKDOWN	..NOT 0, GOTO CHECK DOWN
2AD2 99;	0782	GHI MP	
2AD3 C22963;	0783	LBZ EXDTSC	..IF 0, EXIT DATA SCALE
2AD6 ;	0784		
2AD6 ;	0785	..CHECK BOUNDARY OK FLAG	
2AD6 ;	0786	CKDOWN:	..CHECK BOUNDARY DOWN
2AD6 0C;	0787	LDN ZAP1	
2AD7 CA2ABD;	0788	LBNZ SHDNON	..IF BOUNDARY OK,
2ADA ;	0789		..GOTO SHIFT AGAIN
2ADA ;	0790		
2ADA ;	0791	..ELSE, DECREMENT FREE LENGTH COUNTER	
2ADA 2A;	0792	DEC ZAP2	
2ADB 2A;	0793	DEC ZAP2	
2ADC ;	0794		
2ADC ;	0795	..IF NOT ZERO, GOTO SHIFT AGAIN	
2ADC 8A;	0796	GLO ZAP2	
2ADD CA2ABD;	0797	LBNZ SHDNON	
2AE0 9A;	0798	GHI ZAP2	
2AE1 CA2ABD;	0799	LBNZ SHDNON	
2AE4 ;	0800		
2AE4 ;	0801	..ELSE, SET R(MA) TO FIFO FWA	
2AE4 E8;	0802	SEX FRP	
2AE5 72;	0803	LDXA	
2AE6 BD;	0804	PHI MA	
2AE7 F0;	0805	LDX	
2AE8 AD;	0806	PLO MA	
2AE9 ;	0807		
2AE9 ;	0808	..SET BOUNDARY OK FLAG TRUE	
2AE9 F8FF;	0809	LDI #FF	
2AEB 5C;	0810	STR ZAP1	
2AEC ;	0811		
2AEC ;	0812	..AND SHIFT AGAIN	
2AEC C02ABD;	0813	LBR SHDNON	
2AEF ;	0814		
2AEF ;	0815		

ASP FFT Program  
Program Code

2AEF ;	0816 SHDNMR:	..SHIFT DOWN MORE
2AEF ;	0817 ..MOVE WORKING SHIFT VALUE INTO R(ZAP3)	
2AEF 9E;	0818 GHI MQ	
2AF0 BB;	0819 PHI ZAP3	
2AF1 8E;	0820 GLO MQ	
2AF2 AB;	0821 PLO ZAP3	
2AF3 ;	0822	
2AF3 ;	0823 ..NOW GET DATA WORD	
2AF3 ED;	0824 SEX MA	
2AF4 7Z;	0825 LDXA	
2AF5 BF;	0826 PHI AC	
2AF6 FO;	0827 LDX	
2AF7 AF;	0828 PLO AC	
2AF8 ;	0829	
2AF8 ;	0830 ..NOW SHIFT IT RIGHT (WITH SIGN EXTEND)	
2AF8 9F;	0831 NOWSHR: GHI AC	
2AF9 FE;	0832 SHL	
2AFA CB2AFF;	0833 LBNF DMRRSR	
2AFD F901;	0834 ORI #01	
2AFF ;	0835 DMRRSR:	..DOWN MORE RING SHIFT RIGHT
2AFF 76;	0836 RSHR	
2B00 76;	0837 RSHR	
2B01 BF;	0838 PHI AC	
2B02 8F;	0839 GLO AC	
2B03 76;	0840 SHRC	
2B04 AF;	0841 PLO AC	
2B05 ;	0842	
2B05 ;	0843 ..ENOUGH SHIFTS ?	
2B05 1B;	0844 INC ZAP3	
2B06 8B;	0845 GLO ZAP3	
2B07 CA2AF8;	0846 LBNZ NOWSHR	..IF NOT, SHIFT AGAIN
2B0A ;	0847	
2B0A ;	0848 ..THEN STORE BACK	
2B0A 8F;	0849 GLO AC	
2B0B 73;	0850 STXD	
2B0C 9F;	0851 GHI AC	
2B0D 5D;	0852 STR MA	
2B0E ;	0853	
2B0E ;	0854 ..CHECK FOR DONE	
2B0E 29;	0855 DEC MP	..DECREMENT LENGTH COUNT
2B0F 29;	0856 DEC MP	
2B10 89;	0857 GLO MP	
2B11 CA2B18;	0858 LBNZ NXDNMR	..NOT 0, GOTO NEXT UP MORE
2B14 99;	0859 GHI MP	
2B15 C22963;	0860 LBZ EXDTSC	..IF 0, EXIT DATA SCALE
2B18 ;	0861	
2B18 ;	0862 ..ELSE, GET TO NEXT MEMORY	
2B18 ;	0863 NXDNMR:	..NEXT MEMORY UP MORE
2B18 1D;	0864 INC MA	
2B19 1D;	0865 INC MA	
2B1A ;	0866	



```

2B1A ;          0867 ..CHECK BOUNDARY OK FLAG
2B1A 0C;        0868         LDN ZAP1
2B1B CA2AEF;    0869         LBNZ SHDNMR         ..IF BOUNDARY OK,
2B1E ;          0870         ..GOTO SHIFT AGAIN
2B1E ;          0871
2B1E ;          0872 ..ELSE, DECREMENT FREE LENGTH COUNTER
2B1E 2A;        0873         DEC ZAP2
2B1F 2A;        0874         DEC ZAP2
2B20 ;          0875
2B20 ;          0876 ..IF NOT ZERO, GOTO SHIFT AGAIN
2B20 8A;        0877         GLO ZAP2
2B21 CA2AEF;    0878         LBNZ SHDNMR
2B24 9A;        0879         GHI ZAP2
2B25 CA2AEF;    0880         LBNZ SHDNMR
2B28 ;          0881
2B28 ;          0882 ..ELSE, SET R(MA) TO FIFO FWA
2B28 E8;        0883         SEX FRP
2B29 72;        0884         LDXA
2B2A BD;        0885         PHI MA
2B2B F0;        0886         LDX
2B2C AD;        0887         PLO MA
2B2D ;          0888
2B2D ;          0889 ..SET BOUNDARY OK FLAG TRUE
2B2D F8FF;      0890         LDI #FF
2B2F 5C;        0891         STR ZAP1
2B30 ;          0892
2B30 ;          0893 ..AND SHIFT AGAIN
2B30 C02AEF;    0894         LBR SHDNMR
2B33 ;          0895
2B33 ;          0896
2B33 ;          0897 ..#####
2B33 ;          0898 ..END OF DATA SCALE

```

ASP FFT Program  
Program Code

A.2.3.3 SUBRT2 (Subroutine Block #2)

```

0000 ;          0001
0000 ;          0002 ..SUBRT2.SR          20 JAN 80          2:30 PM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..SUBRT2.SR          SUBROUTINE BLOCK #2
0000 ;          0244 ..#####
0000 ;          0245
0000 ;          0246
0000 ;          0247 ..VECTORS TO INTERNAL ROUTINES
0000 ;          0248
0000 ;          0249          ORG #2B50
2B50 C4C4C4C4; 0250          ,#C4C4C4C4
2B54 3058;     0251          BR BDRYST          ..BOUNDARY SET
2B56 308E;     0252          BR BDRYCK          ..BOUNDARY CHECK
2B58 ;         0253
2B58 ;         0254
2B58 ;         0255 ..#####
2B58 ;         0256 ..BDRYST/CK.SR          BOUNDARY SET/CHECK
2B58 ;         0257 ..
2B58 ;         0258 ..THERE ARE TWO ENTRY POINTS TO THIS ROUTINE
2B58 ;         0259 ..THE FIRST COMPUTES THE LENGTH FROM FIRST ADDRESS
2B58 ;         0260 ..          TO LWA+1, THAT IS FREE LENGTH, AND
2B58 ;         0261 ..          COMPARES IT TO MOVE LENGTH FOR POSSIBLE
2B58 ;         0262 ..          BOUNDARY OK CONDITION.
2B58 ;         0263 ..          IT SETS BDRYOK, FREELN, FWA & MOVECT.
2B58 ;         0264 ..          IT REQUIRES THAT R(MA) HAVE START ADDRESS
2B58 ;         0265 ..          (IT IS LEFT UNCHANGED) AND R(MP) MUST POINT
2B58 ;         0266 ..          TO MOVE COUNT (IT IS INCREMENTED BY 2).
2B58 ;         0267 ..          R(ZAP1) MUST POINT TO LWA+1
2B58 ;         0268 ..          M(R(ZAP1)+2) MUST BE FWA
2B58 ;         0269 ..          R(ZAP1), R(FRP) & R(ZAP2) ARE USED
2B58 ;         0270 ..THE SECOND ENTRY POINT DECREMENTS FREELN AND
2B58 ;         0271 ..          CHECKS FOR LWA+1 (IF REQUIRED).
2B58 ;         0272 ..          IF NECESSARY IT RESETS TO FWA.
2B58 ;         0273 ..          IT THEN DECREMENTS MOVECT TO CHECK FOR DONE
2B58 ;         0274 ..          IF DONE IT RETURNS WITH D = #00
2B58 ;         0275 ..          ELSE D.NOT.#00
2B58 ;         0276 ..          HERE ONLY R(ZAP1) & R(ZAP2) ARE USED
2B58 ;         0277 ..#####
2B58 ;         0278
2B58 ;         0279

```

```

2B58 ;      0280 ..ENTER HERE FOR BOUNDARY SET
2B58 ;      0281 BDRYST:                ..BOUNDARY SET
2B58 ;      0282
2B58 ;      0283 ..COMPUTE FREELN = LWA+1 - START ADDRESS
2B58 EC;     0284         SEX ZAP1
2B59 1C8DF5AA; 0285         INC ZAP1; GLO MA; SD; PLO ZAP2
2B5D 2C9D75BA; 0286         DEC ZAP1; GHI MA; SDB; PHI ZAP2
2B61 ;      0287
2B61 ;      0288 ..STORE FREELN
2B61 F8C6A8;  0289         LDI A.0(FREELN); PLO FRP
2B64 9A5818;  0290         GHI ZAP2; STR FRP; INC FRP
2B67 8A5818;  0291         GLO ZAP2; STR FRP; INC FRP
2B6A ;      0292
2B6A ;      0293 ..STORE FWA VALUE
2B6A 1C1C;    0294         INC ZAP1; INC ZAP1
2B6C 725818;  0295         LDXA; STR FRP; INC FRP
2B6F 725818;  0296         LDXA; STR FRP; INC FRP
2B72 ;      0297
2B72 ;      0298 ..STORE MOVECT
2B72 495818;  0299         LDA MP; STR FRP; INC FRP
2B75 4958;    0300         LDA MP; STR FRP
2B77 ;      0301
2B77 ;      0302 ..CHECK FOR MOVELN <= FREELN
2B77 E88AF7;  0303         SEX FRP; GLO ZAP2; SM
2B7A 289A77;  0304         DEC FRP; GHI ZAP2; SMB
2B7D 3B87;    0305         BNF BDMOVE          ..IF MOVE > FREE, THEN
2B7F ;      0306                     ..GOTO BAD MOVE
2B7F ;      0307
2B7F ;      0308 ..ELSE GOOD MOVE, SET BOUNDARY OK FLAG TRUE (#00)
2B7F F8C5AC;  0309         LDI A.0(BDRYOK); PLO ZAP1
2B82 F8005C;  0310         LDI #00; STR ZAP1
2B85 308D;    0311         BR EXBDSC          ..AND BRANCH OUT
2B87 ;      0312
2B87 ;      0313 BDMOVE:                ..BAD MOVE
2B87 ;      0314 ..SET BOUNDARY OK FLAG FALSE (#FF)
2B87 F8C5AC;  0315         LDI A.0(BDRYOK); PLO ZAP1
2B8A F8FF5C;  0316         LDI #FF; STR ZAP1
2B8D ;      0317
2B8D ;      0318 ..AND FALL OUT
2B8D ;      0319
2B8D ;      0320 ..-----
2B8D ;      0321 EXBDSC:                ..EXIT BOUNDARY SET/CHECK
2B8D ;      0322 ..RETURN TO MAIN PROGRAM
2B8D D3;      0323         SEP PC
2B8E ;      0324
2B8E ;      0325

```

ASP FFT Program  
Program Code

```

2B8E ;          0326 ..*****
2B8E ;          0327 ..BDRYCK.SR          BOUNDARY CHECK
2B8E ;          0328 ..
2B8E ;          0329 ..ENTER HERE TO CHECK FOR LWA+1 CONDITION
2B8E ;          0330 ..THEN CHECK FOR DONE WITH BLOCK
2B8E ;          0331 ..R(MA) IS THE ADDRESS POINTER IN QUESTION
2B8E ;          0332 ..R(ZAP1) & R(ZAP2) ARE USED IN THE ROUTINUE
2B8E ;          0333 ..*****
2B8E ;          0334
2B8E ;          0335
2B8E ;          0336 ..ENTER HERE FOR BOUNDARY CHECK
2B8E ;          0337 BDRYCK:          ..BOUNDARY CHECK
2B8E ;          0338
2B8E ;          0339 ..CHECK FOR BOUNDARY OK
2B8E F8C5AC;    0340          LDI A.0(BDRYOK); PLO ZAP1
2B91 EC72;      0341          SEX ZAP1; LDXA
2B93 32B0;      0342          BZ CKMVDN          ..IF BOUNDARY OK, THEN
2B95 ;          0343          ..GOTO CHECK MOVE DONE
2B95 ;          0344
2B95 ;          0345 ..ELSE COUNT DOWN FREE LENGTH BY 2
2B95 1C;        0346          INC ZAP1
2B96 F802F573AA; 0347          LDI #02; SD; STXD; PLO ZAP2
2B9B F800755C;  0348          LDI #00; SDB; STR ZAP1
2B9F ;          0349
2B9F ;          0350 ..CHECK FOR FREE LENGTH AT #0000
2B9F 3AB0;      0351          BNZ CKMVDN
2BA1 8A3AB0;    0352          GLO ZAP2; BNZ CKMVDN ..IF NOT #0000, THEN
2BA4 ;          0353          ..GOTO CHECK MOVE DONE
2BA4 ;          0354
2BA4 ;          0355 ..ELSE AT LWA+1, RESET ADDRESS POINTER TO FWA
2BA4 1C1C;      0356          INC ZAP1; INC ZAP1
2BA6 72BD72AD;  0357          LDXA; PHI MA; LDXA; PLO MA
2BAA ;          0358
2BAA ;          0359 ..AND SET BOUNDARY FLAG TRUE (#00)
2BAA F8C5AC;    0360          LDI A.0(BDRYOK); PLO ZAP1
2BAD F8005C;    0361          LDI #00; STR ZAP1
2BBO ;          0362
2BBO ;          0363 CKMVDN:          ..CHECK MOVE DONE
2BBO ;          0364 ..COUNT DOWN MOVE COUNT BY 2
2BBO F8CBAC;    0365          LDI A.0(MOVECT+1); PLO ZAP1
2BB3 F802F573AA; 0366          LDI #02; SD; STXD; PLO ZAP2
2BB8 F8007573;  0367          LDI #00; SDB; STXD
2BBC ;          0368
2BBC ;          0369 ..CHECK FOR DONE WITH MOVE
2BBC ;          0370 ..CHECK HIGH BYTE, IF NOT #00
2BBC ;          0371 ..RETURN WITH NOT DONE FLAG (NOT #00)
2BBC 3A8D;      0372          BNZ EXBDSC
2BBE ;          0373

```



ASP FFT Program  
Program Code

```

2BBE ; 0374 ..CHECK LOW BYTE AND RETURN WITH FLAG EITHER
2BBE ; 0375 ..NOT #00, NOT DONE
2BBE ; 0376 ..OR IS #00, IS DONE
2BBE 8A308D; 0377 GLO ZAP2; BR EXBDSC
2BC1 ; 0378
2BC1 ; 0379
2BC1 ; 0380 ..#####
2BC1 ; 0381 ..END OF SUBROUTINE BLOCK #2
2BC1 ; 0382

```

ASP FFT Program  
Program Code

A.2.4 Constants and Tables

A.2.4.1 SYSSET (System Set)

```

0000 ;          0001
0000 ;          0002 ..SYSSET.SR          12 JAN 80          12:45 PM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..SYSSET.SR          SYSTEM SET
0000 ;          0244 ..
0000 ;          0245 ..THIS PROGRAM SETS THE FFT ROM TABLES:
0000 ;          0246 ..      LENGTH SET TABLES,
0000 ;          0247 ..      P & S SHAPE TABLES,
0000 ;          0248 ..      TRIG & SHAPING TABLE DIRECTORIES, AND
0000 ;          0249 ..      THE BIT MASK TABLE
0000 ;          0250 ..      FFT LENGTH TABLES
0000 ;          0251 ..AND FOR RIGHT NOW, SETS YP & YS TABLE DATA
0000 ;          0252 ..      FOR THE YPFIFO & YSFIFO
0000 ;          0253 ..#####
0000 ;          0254
0000 ;          0255
0000 ;          0256          ORG FFTBL
2660 ;          0257
2660 ;          0258 ..BEGIN HERE
2660 ;          0259
2660 ;          0260
2660 ;          0261 ..***** LENGTH SET TABLES *****
2660 ;          0262
2660 7FFF;          0263          ,#7FFF          ..NO S DEFAULT
2662 0100;          0264          ,#0100          ..PLN=128
2664 0200;          0265          ,#0200          ..SLN=256
2666 268E;          0266          ,A(PLN128)
2668 26AA;          0267          ,A(SLN256)
266A ;          0268
266A 0080;          0269          ,#0080          ..DT < 128
266C 0080;          0270          ,#0080          ..PLN=64
266E 0100;          0271          ,#0100          ..SLN=128
2670 2688;          0272          ,A(PLN64)
2672 26A0;          0273          ,A(SLN128)
2674 ;          0274
2674 0258;          0275          ,#0258          ..DT < 600
2676 0100;          0276          ,#0100          ..PLN=128
2678 0200;          0277          ,#0200          ..SLN=256
267A 268E;          0278          ,A(PLN128)
267C 26AA;          0279          ,A(SLN256)
267E ;          0280

```

267E 7FFF;	0281	,#7FFF	..DT < MAXPOS
2680 0200;	0282	,#0200	..PLN=256
2682 0400;	0283	,#0400	..SLN=512
2684 2694;	0284	,A(PLN256)	
2686 26B4;	0285	,A(SLN512)	
2688 ;	0286		
2688 ;	0287		
2688 ;	0288	..***** P & S SHAPE TABLES *****	
2688 ;	0289		
2688 ;	0290	..P-FFT LENGTH = 64	
2688 0060;	0291	,#0060	..75% COUNT
268A 0020;	0292	,#0020	..25% COUNT
268C 0020;	0293	,#0020	..SKIP=512/16
268E ;	0294		
268E ;	0295	..P-FFT LENGTH = 128	
268E 00C0;	0296	,#00C0	
2690 0040;	0297	,#0040	
2692 0010;	0298	,#0010	
2694 ;	0299		
2694 ;	0300	..P-FFT LENGTH = 256	
2694 0180;	0301	,#0180	
2696 0080;	0302	,#0080	
2698 0008;	0303	,#0008	
269A ;	0304		
269A ;	0305	..P-FFT LENGTH = 512	
269A 0300;	0306	,#0300	
269C 0100;	0307	,#0100	
269E 0004;	0308	,#0004	
26A0 ;	0309		
26A0 ;	0310	..S-FFT LENGTH = 128	
26A0 0020;	0311	,#0020	..12.5% HEAD COUNT
26A2 0020;	0312	,#0020	..HEAD SKIP=512/16
26A4 00A0;	0313	,#00A0	..62.5% MIDDLE COUNT
26A6 0040;	0314	,#0040	..25% TAIL COUNT
26A8 0010;	0315	,#0010	..TAIL SKIP=512/32
26AA ;	0316		
26AA ;	0317	..S-FFT LENGTH = 256	
26AA 0040;	0318	,#0040	
26AC 0010;	0319	,#0010	
26AE 0140;	0320	,#0140	
26B0 0080;	0321	,#0080	
26B2 0008;	0322	,#0008	
26B4 ;	0323		
26B4 ;	0324	..S-FFT LENGTH = 512	
26B4 0080;	0325	,#0080	
26B6 0008;	0326	,#0008	
26B8 0280;	0327	,#0280	
26BA 0100;	0328	,#0100	
26BC 0004;	0329	,#0004	
26BE ;	0330		
26BE ;	0331		

ASP FFT Program  
Program Code

```

26BE ;          0332 ..***** TRIG TABLE DIRECTORY *****
26BE ;          0333
26BE 27FE;      0334          ,A(TRGLWA)
26C0 26FE;      0335          ,A(TRGFWA)          ..TRGDIR
26C2 ;          0336
26C2 ;          0337
26C2 ;          0338 ..***** BIT MASK TABLE *****
26C2 ;          0339
26C2 0400;      0340          ,#0400
26C4 0200;      0341          ,#0200
26C6 0100;      0342          ,#0100
26C8 0080;      0343          ,#0080
26CA 0040;      0344          ,#0040
26CC 0020;      0345          ,#0020
26CE 0010;      0346          ,#0010
26D0 0008;      0347          ,#0008
26D2 0004;      0348          ,#0004          ..VALID COLUMN COMPUTATIONS
26D4 0000;      0349          ,#0000          ..FFT END FLAG
26D6 ;          0350
26D6 ;          0351
26D6 ;          0352 ..***** FFT LENGTH TABLES *****
26D6 ;          0353
26D6 26C2;      0354          ,A(BITBTL)          ..FLN512
26D8 000C;      0355          ,#000C          ..POINTER TO BIT MASK
26DA 0002;      0356          ,#0002          ..INC/DEC VALUE FOR ORDER
26DC 0000;      0357          ,#0000
26DE ;          0358
26DE 26C4;      0359          ,A(BITBTL+2)          ..FLN256
26E0 000B;      0360          ,#000B
26E2 0004;      0361          ,#0004
26E4 0000;      0362          ,#0000
26E6 ;          0363
26E6 26C6;      0364          ,A(BITBTL+4)          ..FLN128
26E8 000A;      0365          ,#000A
26EA 0008;      0366          ,#0008
26EC 0000;      0367          ,#0000
26EE ;          0368
26EE 26C8;      0369          ,A(BITBTL+6)          ..FLN64
26F0 0009;      0370          ,#0009
26F2 0010;      0371          ,#0010
26F4 0000;      0372          ,#0000
26F6 ;          0373
26F6 ;          0374

```



ASP FFT Program  
Program Code

```

26F6 ;      0375 ..***** YP & YS TABLE DATA *****
26F6 ;      0376
26F6 ;      0377      ORG YPTBL
30A5 ;      0378
30A5 3601;  0379      ,A(YPFWA)
30A7 3A01;  0380      ,A(YPLWA+1)
30A9 3601;  0381      ,A(YPFWA)
30AB 0000;  0382      ,#0000
30AD 00;    0383      ,#00      ..YPFIFO
30AE ;      0384
30AE 3A01;  0385      ,A(YSFWA)
30B0 3E01;  0386      ,A(YSLWA+1)
30B2 3A01;  0387      ,A(YSFWA)
30B4 0000;  0388      ,#0000
30B6 00;    0389      ,#00      ..YSFIFO
30B7 ;      0390
30B7 ;      0391
30B7 ;      0392      END

```

ASP FFT Program  
Program Code

A.2.4.2 TRGTBL (Trig Table)

```

0000 ;          0001
0000 ;          0002 ..TRGTBL.SR          12 JAN 80          1:20 PM
0000 ;          0003
0000 ;          0004
0000 ;          0005 ..*****
0000 ;          0006 ..          TRGTBL.SR
0000 ;          0007 ..
0000 ;          0008 ..TRGTBL IS A LISTING OF HEX SINE VALUES
0000 ;          0009 .. SIN (2*PI*K/512) FOR K = 0 TO 128
0000 ;          0010 ..RANGING FROM 0 HEX 0000)
0000 ;          0011 .. TO ALMOST +1 (HEX 7FFF)
0000 ;          0012 ..*****
0000 ;          0013
0000 ;          0014
0000 ;          0015          ORG #26FE
26FE ;          0016
26FE ;          0017 TRGFWA:          ..SIN
26FE 0000;          0018          ,#0000          ..0
2700 0192;          0019          ,#0192
2702 0324;          0020          ,#0324
2704 04B6;          0021          ,#04B6
2706 0648;          0022          ,#0648
2708 07D9;          0023          ,#07D9
270A 096B;          0024          ,#096B
270C 0AFB;          0025          ,#0AFB
270E 0C8C;          0026          ,#0C8C
2710 0E1C;          0027          ,#0E1C
2712 0FAB;          0028          ,#0FAB
2714 113A;          0029          ,#113A
2716 12C8;          0030          ,#12C8
2718 1455;          0031          ,#1455
271A 15E2;          0032          ,#15E2
271C 176E;          0033          ,#176E
271E 18F9;          0034          ,#18F9          ..16
2720 1A83;          0035          ,#1A83
2722 1C0C;          0036          ,#1C0C
2724 1D93;          0037          ,#1D93
2726 1F1A;          0038          ,#1F1A
2728 209F;          0039          ,#209F
272A 2224;          0040          ,#2224
272C 23A7;          0041          ,#23A7
272E 2528;          0042          ,#2528
2730 26A8;          0043          ,#26A8
2732 2827;          0044          ,#2827
2734 29A4;          0045          ,#29A4

```

ASP FFT Program  
Program Code

2736 2B1F;	0046	,#2B1F	
2738 2C99;	0047	,#2C99	
273A 2E11;	0048	,#2E11	
273C 2F87;	0049	,#2F87	
273E 30FC;	0050	,#30FC	..32
2740 326E;	0051	,#326E	
2742 33DF;	0052	,#33DF	
2744 354E;	0053	,#354E	
2746 36BA;	0054	,#36BA	
2748 3825;	0055	,#3825	
274A 398D;	0056	,#398D	
274C 3AF3;	0057	,#3AF3	
274E 3C57;	0058	,#3C57	
2750 3DB8;	0059	,#3DB8	
2752 3F17;	0060	,#3F17	
2754 4074;	0061	,#4074	
2756 41CE;	0062	,#41CE	
2758 4326;	0063	,#4326	
275A 447B;	0064	,#447B	
275C 45CD;	0065	,#45CD	
275E 471D;	0066	,#471D	..48
2760 486A;	0067	,#486A	
2762 49B4;	0068	,#49B4	
2764 4AFB;	0069	,#4AFB	
2766 4C40;	0070	,#4C40	
2768 4D81;	0071	,#4D81	
276A 4ECO;	0072	,#4ECO	
276C 4FFB;	0073	,#4FFB	
276E 5134;	0074	,#5134	
2770 5269;	0075	,#5269	
2772 539B;	0076	,#539B	
2774 54CA;	0077	,#54CA	
2776 55F6;	0078	,#55F6	
2778 571E;	0079	,#571E	
277A 5843;	0080	,#5843	
277C 5964;	0081	,#5964	
277E 5A82;	0082	,#5A82	..64
2780 5B9D;	0083	,#5B9D	
2782 5CB4;	0084	,#5CB4	
2784 5DC8;	0085	,#5DC8	
2786 5ED7;	0086	,#5ED7	
2788 5FE4;	0087	,#5FE4	
278A 60EC;	0088	,#60EC	
278C 61F1;	0089	,#61F1	
278E 62F2;	0090	,#62F2	
2790 63EF;	0091	,#63EF	
2792 64E9;	0092	,#64E9	
2794 65DE;	0093	,#65DE	
2796 66D0;	0094	,#66D0	
2798 67BD;	0095	,#67BD	
279A 68A7;	0096	,#68A7	

ASP FFT Program  
Program Code

279C 698C;	0097	,#698C	
279E 6A6E;	0098	,#6A6E	..80
27A0 6B4B;	0099	,#6B4B	
27A2 6C24;	0100	,#6C24	
27A4 6CF9;	0101	,#6CF9	
27A6 6DCA;	0102	,#6DCA	
27A8 6E97;	0103	,#6E97	
27AA 6F5F;	0104	,#6F5F	
27AC 7023;	0105	,#7023	
27AE 70E3;	0106	,#70E3	
27B0 719E;	0107	,#719E	
27B2 7255;	0108	,#7255	
27B4 7308;	0109	,#7308	
27B6 73B6;	0110	,#73B6	
27B8 7460;	0111	,#7460	
27BA 7505;	0112	,#7505	
27BC 75A6;	0113	,#75A6	
27BE 7642;	0114	,#7642	..96
27C0 76D9;	0115	,#76D9	
27C2 776C;	0116	,#776C	
27C4 77FB;	0117	,#77FB	
27C6 7885;	0118	,#7885	
27C8 790A;	0119	,#790A	
27CA 798A;	0120	,#798A	
27CC 7A06;	0121	,#7A06	
27CE 7A7D;	0122	,#7A7D	
27D0 7AEF;	0123	,#7AEF	
27D2 7B5D;	0124	,#7B5D	
27D4 7BC6;	0125	,#7BC6	
27D6 7C2A;	0126	,#7C2A	
27D8 7C89;	0127	,#7C89	
27DA 7CE4;	0128	,#7CE4	
27DC 7D3A;	0129	,#7D3A	
27DE 7D8A;	0130	,#7D8A	..112
27E0 7DD6;	0131	,#7DD6	
27E2 7E1E;	0132	,#7E1E	
27E4 7E60;	0133	,#7E60	
27E6 7E9D;	0134	,#7E9D	
27E8 7ED6;	0135	,#7ED6	
27EA 7F0A;	0136	,#7F0A	
27EC 7F38;	0137	,#7F38	
27EE 7F62;	0138	,#7F62	
27F0 7F87;	0139	,#7F87	
27F2 7FA7;	0140	,#7FA7	
27F4 7FC2;	0141	,#7FC2	
27F6 7FD9;	0142	,#7FD9	
27F8 7FEA;	0143	,#7FEA	
27FA 7FF6;	0144	,#7FF6	
27FC 7FFE;	0145	,#7FFE	
27FE 7FFF;	0146	TRGLWA: ,#7FFF	..128
2800 ;	0147		



### A.2.4.3 LOGTBL (Logarithm Table)

```

0000 ;          0001
0000 ;          0002 ..*****
0000 ;          0003 ..LOGTBL.SR          22 FEB 80          9:50 PM
0000 ;          0004 ..
0000 ;          0005 ..THIS IS A TABLE OF MANTISSA VALUES, BASE 2, FOR
0000 ;          0006 ..
0000 ;          0007 .. LOG(1 + (N/128) + (1/256))    FOR N = [0...127]
0000 ;          0008 ..
0000 ;          0009 ..NOTE: THE 1/256 FACTOR YIELDS THE MEDIAN FOR
0000 ;          0010 ..      THE INCREMENT FROM ONE N TO THE NEXT N
0000 ;          0011 ..*****
0000 ;          0012
0000 ;          0013          ORG #2D00
0000 ;          0014
0000 ;          0015 LGBASE:          ..LOGBASE
2D00 00B8;          0016          ,#00B8          ..0
2D02 0227;          0017          ,#0227
2D04 0392;          0018          ,#0392
2D06 04FB;          0019          ,#04FB
2D08 0661;          0020          ,#0661
2D0A 07C5;          0021          ,#07C5
2D0C 0926;          0022          ,#0926
2D0E 0A84;          0023          ,#0A84
2D10 0BDF;          0024          ,#0BDF
2D12 0D39;          0025          ,#0D39
2D14 0E8F;          0026          ,#0E8F
2D16 0FE3;          0027          ,#0FE3
2D18 1135;          0028          ,#1135
2D1A 1284;          0029          ,#1284
2D1C 13D1;          0030          ,#13D1
2D1E 151C;          0031          ,#151C
2D20 1664;          0032          ,#1664          ..16
2D22 17AA;          0033          ,#17AA
2D24 18EE;          0034          ,#18EE
2D26 1A2F;          0035          ,#1A2F
2D28 1B6F;          0036          ,#1B6F
2D2A 1CAC;          0037          ,#1CAC
2D2C 1DE7;          0038          ,#1DE7
2D2E 1F20;          0039          ,#1F20
2D30 2057;          0040          ,#2057
2D32 218C;          0041          ,#218C
2D34 22BF;          0042          ,#22BF
2D36 23F0;          0043          ,#23F0
2D38 251F;          0044          ,#251F
2D3A 264C;          0045          ,#264C

```

ASP FFT Program  
Program Code

2D3C 2778;	0046	,#2778	
2D3E 28A1;	0047	,#28A1	
2D40 29C8;	0048	,#29C8	..32
2D42 2AEE;	0049	,#2AEE	
2D44 2C12;	0050	,#2C12	
2D46 2D34;	0051	,#2D34	
2D48 2E54;	0052	,#2E54	
2D4A 2F73;	0053	,#2F73	
2D4C 308F;	0054	,#308F	
2D4E 31AB;	0055	,#31AB	
2D50 32C4;	0056	,#32C4	
2D52 33DC;	0057	,#33DC	
2D54 34F2;	0058	,#34F2	
2D56 3606;	0059	,#3606	
2D58 3719;	0060	,#3719	
2D5A 382A;	0061	,#382A	
2D5C 393A;	0062	,#393A	
2D5E 3A48;	0063	,#3A48	
2D60 3B55;	0064	,#3B55	..48
2D62 3C60;	0065	,#3C60	
2D64 3D69;	0066	,#3D69	
2D66 3E72;	0067	,#3E72	
2D68 3F78;	0068	,#3F78	
2D6A 407D;	0069	,#407D	
2D6C 4181;	0070	,#4181	
2D6E 4283;	0071	,#4283	
2D70 4384;	0072	,#4384	
2D72 4484;	0073	,#4484	
2D74 4582;	0074	,#4582	
2D76 467F;	0075	,#467F	
2D78 477A;	0076	,#477A	
2D7A 4874;	0077	,#4874	
2D7C 496D;	0078	,#496D	
2D7E 4A65;	0079	,#4A65	
2D80 4B5B;	0080	,#4B5B	..64
2D82 4C50;	0081	,#4C50	
2D84 4D44;	0082	,#4D44	
2D86 4E36;	0083	,#4E36	
2D88 4F27;	0084	,#4F27	
2D8A 5017;	0085	,#5017	
2D8C 5106;	0086	,#5106	
2D8E 51F4;	0087	,#51F4	
2D90 52E0;	0088	,#52E0	
2D92 53CB;	0089	,#53CB	
2D94 54B5;	0090	,#54B5	
2D96 559E;	0091	,#559E	
2D98 5686;	0092	,#5686	
2D9A 576C;	0093	,#576C	
2D9C 5852;	0094	,#5852	
2D9E 5936;	0095	,#5936	
2DA0 5A1A;	0096	,#5A1A	..80

2DA2 5AFC;	0097	,#5AFC	
2DA4 5BDD;	0098	,#5BDD	
2DA6 5CBD;	0099	,#5CBD	
2DA8 5D9C;	0100	,#5D9C	
2DAA 5E7A;	0101	,#5E7A	
2DAC 5F57;	0102	,#5F57	
2DAE 6033;	0103	,#6033	
2DB0 610D;	0104	,#610D	
2DB2 61E7;	0105	,#61E7	
2DB4 62C0;	0106	,#62C0	
2DB6 6398;	0107	,#6398	
2DB8 646F;	0108	,#646F	
2DBA 6545;	0109	,#6545	
2DBC 661A;	0110	,#661A	
2DBE 66EE;	0111	,#66EE	
2DC0 67C1;	0112	,#67C1	..96
2DC2 6893;	0113	,#6893	
2DC4 6964;	0114	,#6964	
2DC6 6A34;	0115	,#6A34	
2DC8 6B04;	0116	,#6B04	
2DCA 6BD2;	0117	,#6BD2	
2DCC 6CA0;	0118	,#6CA0	
2DCE 6D6C;	0119	,#6D6C	
2DD0 6E38;	0120	,#6E38	
2DD2 6F03;	0121	,#6F03	
2DD4 6FCD;	0122	,#6FCD	
2DD6 7096;	0123	,#7096	
2DD8 715E;	0124	,#715E	
2DDA 7226;	0125	,#7226	
2DDC 72ED;	0126	,#72ED	
2DDE 73B2;	0127	,#73B2	
2DE0 7477;	0128	,#7477	..112
2DE2 753C;	0129	,#753C	
2DE4 75FF;	0130	,#75FF	
2DE6 76C1;	0131	,#76C1	
2DE8 7783;	0132	,#7783	
2DEA 7844;	0133	,#7844	
2DEC 7904;	0134	,#7904	
2DEE 79C4;	0135	,#79C4	
2DF0 7A82;	0136	,#7A82	
2DF2 7B40;	0137	,#7B40	
2DF4 7BFD;	0138	,#7BFD	
2DF6 7CBA;	0139	,#7CBA	
2DF8 7D75;	0140	,#7D75	
2DFA 7E30;	0141	,#7E30	
2DFC 7EEA;	0142	,#7EEA	
2DFA 7FA4;	0143	,#7FA4	..127
2E00 ;	0144	..#####	
2E00 ;	0145	..END OF LOG TABLE.SR	

ASP FFT Program  
Program Code

A.2.4.4 INSFIX (Instrument Fix)

```

0000 ;          0001
0000 ;          0002 ..INSFIX.SR          26 MAY 80          5:50 PM
0000 ;          0003
0000 ;          0004 ..*****
0000 ;          0005 ..THIS IS A TABLE OF INSTRUMENT CORRECTION VALUES
0000 ;          0006 ..FOR A GEOPHONE OPERATING AT 100 HZ.
0000 ;          0007 ..IT IS NORMALIZED FOR N=26 @ 5.0781 HZ
0000 ;          0008 ..THUS FOR THE ENERGY MAGNITUDE COMPUTATION YOU
0000 ;          0009 ..NEED TO REMULTIPLY BY N=26'S ACTUAL MAGNITUDE
0000 ;          0010 ..( 3.40005E-2 = #045A * 2**(#0000)
0000 ;          0011 ..          = #45A2 * 2**(#FFFC)
0000 ;          0012 ..*****
0000 ;          0013
0000 ;          0014          ORG #2E00
2E00 ;          0015
2E00 7FFF;      0016          ,#7FFF          ..0
2E02 7FFF;      0017          ,#7FFF
2E04 7FFF;      0018          ,#7FFF
2E06 7FFF;      0019          ,#7FFF
2E08 7FFF;      0020          ,#7FFF
2E0A 7FFF;      0021          ,#7FFF
2E0C 7FFF;      0022          ,#7FFF
2E0E 7FFF;      0023          ,#7FFF
2E10 7FFF;      0024          ,#7FFF
2E12 7FFF;      0025          ,#7FFF
2E14 7FFF;      0026          ,#7FFF
2E16 7FFF;      0027          ,#7FFF
2E18 7FFF;      0028          ,#7FFF
2E1A 7FFF;      0029          ,#7FFF
2E1C 7FFF;      0030          ,#7FFF
2E1E 7FFF;      0031          ,#7FFF
2E20 7FFF;      0032          ,#7FFF          ..16
2E22 7FFF;      0033          ,#7FFF
2E24 7FFF;      0034          ,#7FFF
2E26 7FFF;      0035          ,#7FFF
2E28 7FFF;      0036          ,#7FFF
2E2A 7FFF;      0037          ,#7FFF
2E2C 7FFF;      0038          ,#7FFF
2E2E 7FFF;      0039          ,#7FFF
2E30 7FFF;      0040          ,#7FFF
2E32 7FFF;      0041          ,#7FFF
2E34 7FFF;      0042          ,#7FFF          ..26 (NORMALIZE FACTOR)
2E36 7860;      0043          ,#7860
2E38 71D2;      0044          ,#71D2
2E3A 6C22;      0045          ,#6C22

```



ASP FFT Program  
Program Code

2E3C 6727;	0046	,#6727	
2E3E 62C0;	0047	,#62C0	
2E40 5ED3;	0048	,#5ED3	..32
2E42 5B4D;	0049	,#5B4D	
2E44 581D;	0050	,#581D	
2E46 5535;	0051	,#5535	
2E48 528C;	0052	,#528C	
2E4A 5018;	0053	,#5018	
2E4C 4DD2;	0054	,#4DD2	
2E4E 4BB5;	0055	,#4BB5	
2E50 49BB;	0056	,#49BB	
2E52 47E1;	0057	,#47E1	
2E54 4622;	0058	,#4622	
2E56 447D;	0059	,#447D	
2E58 42EE;	0060	,#42EE	
2E5A 4174;	0061	,#4174	
2E5C 400D;	0062	,#400D	
2E5E 3EB6;	0063	,#3EB6	
2E60 3D6F;	0064	,#3D6F	..48
2E62 3C37;	0065	,#3C37	
2E64 3B0B;	0066	,#3B0B	
2E66 39EC;	0067	,#39EC	
2E68 38D9;	0068	,#38D9	
2E6A 37D0;	0069	,#37D0	
2E6C 36D1;	0070	,#36D1	
2E6E 35DC;	0071	,#35DC	
2E70 34F0;	0072	,#34F0	
2E72 340B;	0073	,#340B	
2E74 332F;	0074	,#332F	
2E76 325A;	0075	,#325A	
2E78 318C;	0076	,#318C	
2E7A 30C5;	0077	,#30C5	
2E7C 3004;	0078	,#3004	
2E7E 2F49;	0079	,#2F49	
2E80 2E93;	0080	,#2E93	..64
2E82 2DE3;	0081	,#2DE3	
2E84 2D38;	0082	,#2D38	
2E86 2C93;	0083	,#2C93	
2E88 2BF2;	0084	,#2BF2	
2E8A 2B55;	0085	,#2B55	
2E8C 2ABD;	0086	,#2ABD	
2E8E 2A28;	0087	,#2A28	
2E90 2998;	0088	,#2998	
2E92 290C;	0089	,#290C	
2E94 2883;	0090	,#2883	
2E96 27FE;	0091	,#27FE	
2E98 277C;	0092	,#277C	
2E9A 26FE;	0093	,#26FE	
2E9C 2682;	0094	,#2682	
2E9E 260A;	0095	,#260A	
2EA0 2594;	0096	,#2594	..80

ASP FFT Program  
Program Code

2EA2 2521;	0097	,#2521	
2EA4 24B1;	0098	,#24B1	
2EA6 2444;	0099	,#2444	
2EA8 23D9;	0100	,#23D9	
2EAA 2370;	0101	,#2370	
2EAC 230A;	0102	,#230A	
2EAE 22A6;	0103	,#22A6	
2EB0 2244;	0104	,#2244	
2EB2 21E4;	0105	,#21E4	
2EB4 2187;	0106	,#2187	
2EB6 212B;	0107	,#212B	
2EB8 20D2;	0108	,#20D2	
2EBA 207A;	0109	,#207A	
2EBC 2024;	0110	,#2024	
2EBE 1FCF;	0111	,#1FCF	
2EC0 1F7D;	0112	,#1F7D	..96
2EC2 1F2C;	0113	,#1F2C	
2EC4 1EDD;	0114	,#1EDD	
2EC6 1E8F;	0115	,#1E8F	
2EC8 1E42;	0116	,#1E42	
2ECA 1DF8;	0117	,#1DF8	
2ECC 1DAE;	0118	,#1DAE	
2ECE 1D66;	0119	,#1D66	
2ED0 1D20;	0120	,#1D20	
2ED2 1CDA;	0121	,#1CDA	
2ED4 1C96;	0122	,#1C96	
2ED6 1C53;	0123	,#1C53	
2ED8 1C11;	0124	,#1C11	
2EDA 1BD1;	0125	,#1BD1	
2EDC 1B92;	0126	,#1B92	
2EDE 1B53;	0127	,#1B53	
2EE0 1B16;	0128	,#1B16	..112
2EE2 1ADA;	0129	,#1ADA	
2EE4 1A9F;	0130	,#1A9F	
2EE6 1A65;	0131	,#1A65	
2EE8 1A2C;	0132	,#1A2C	
2EEA 19F4;	0133	,#19F4	
2EEC 19BC;	0134	,#19BC	
2EEE 1986;	0135	,#1986	
2EF0 1951;	0136	,#1951	
2EF2 191C;	0137	,#191C	
2EF4 18E8;	0138	,#18E8	
2EF6 18B5;	0139	,#18B5	
2EF8 1883;	0140	,#1883	
2EFA 1852;	0141	,#1852	
2EF2 1821;	0142	,#1821	
2EFE 17F1;	0143	,#17F1	
2F00 17C2;	0144	,#17C2	..128
2F02 1794;	0145	,#1794	
2F04 1766;	0146	,#1766	
2F06 1739;	0147	,#1739	

2F08 170D;	0148	,#170D	
2FOA 16E1;	0149	,#16E1	
2FOC 16B6;	0150	,#16B6	
2FOE 168C;	0151	,#168C	
2F10 1662;	0152	,#1662	
2F12 1639;	0153	,#1639	
2F14 1610;	0154	,#1610	
2F16 15E8;	0155	,#15E8	
2F18 15C0;	0156	,#15C0	
2F1A 1599;	0157	,#1599	
2F1C 1573;	0158	,#1573	
2F1E 154D;	0159	,#154D	
2F20 1528;	0160	,#1528	..144
2F22 1503;	0161	,#1503	
2F24 14DF;	0162	,#14DF	
2F26 14BB;	0163	,#14BB	
2F28 1497;	0164	,#1497	
2F2A 1474;	0165	,#1474	
2F2C 1452;	0166	,#1452	
2F2E 1430;	0167	,#1430	
2F30 140E;	0168	,#140E	
2F32 13ED;	0169	,#13ED	
2F34 13CC;	0170	,#13CC	
2F36 13AC;	0171	,#13AC	
2F38 138C;	0172	,#138C	
2F3A 136D;	0173	,#136D	
2F3C 134D;	0174	,#134D	
2F3E 132F;	0175	,#132F	
2F40 1310;	0176	,#1310	..160
2F42 12F2;	0177	,#12F2	
2F44 12D5;	0178	,#12D5	
2F46 12B7;	0179	,#12B7	
2F48 129B;	0180	,#129B	
2F4A 127E;	0181	,#127E	
2F4C 1262;	0182	,#1262	
2F4E 1246;	0183	,#1246	
2F50 122A;	0184	,#122A	
2F52 120F;	0185	,#120F	
2F54 11F4;	0186	,#11F4	
2F56 11D9;	0187	,#11D9	
2F58 11BF;	0188	,#11BF	
2F5A 11A5;	0189	,#11A5	
2F5C 118B;	0190	,#118B	
2F5E 1172;	0191	,#1172	
2F60 1159;	0192	,#1159	..176
2F62 1140;	0193	,#1140	
2F64 1127;	0194	,#1127	
2F66 110F;	0195	,#110F	
2F68 10F7;	0196	,#10F7	
2F6A 10DF;	0197	,#10DF	
2F6C 10C8;	0198	,#10C8	

ASP FFT Program  
Program Code

2F6E 10B0;	0199	,#10B0	
2F70 1099;	0200	,#1099	
2F72 1083;	0201	,#1083	
2F74 106C;	0202	,#106C	
2F76 1056;	0203	,#1056	
2F78 1040;	0204	,#1040	
2F7A 102A;	0205	,#102A	
2F7C 1014;	0206	,#1014	
2F7E OFFF;	0207	,#OFFF	
2F80 OFEA;	0208	,#OFEA	..192
2F82 OFD5;	0209	,#OFD5	
2F84 OFC0;	0210	,#OFC0	
2F86 OFAB;	0211	,#OFAB	
2F88 OF97;	0212	,#OF97	
2F8A OF83;	0213	,#OF83	
2F8C OF6F;	0214	,#OF6F	
2F8E OF5B;	0215	,#OF5B	
2F90 OF48;	0216	,#OF48	
2F92 OF35;	0217	,#OF35	
2F94 OF21;	0218	,#OF21	
2F96 OF0E;	0219	,#OF0E	
2F98 OEFC;	0220	,#OEFC	
2F9A OEE9;	0221	,#OEE9	
2F9C OED7;	0222	,#OED7	
2F9E OEC4;	0223	,#OEC4	
2FA0 OEB2;	0224	,#OEB2	..208
2FA2 OEA1;	0225	,#OEA1	
2FA4 OE8F;	0226	,#OE8F	
2FA6 OE7D;	0227	,#OE7D	
2FA8 OE6C;	0228	,#OE6C	
2FAA OE5B;	0229	,#OE5B	
2FAC OE4A;	0230	,#OE4A	
2FAE OE39;	0231	,#OE39	
2FB0 OE28;	0232	,#OE28	
2FB2 OE17;	0233	,#OE17	
2FB4 OE07;	0234	,#OE07	
2FB6 ODF7;	0235	,#ODF7	
2FB8 ODE6;	0236	,#ODE6	
2FBA ODD6;	0237	,#ODD6	
2FBC ODC7;	0238	,#ODC7	
2FBE ODB7;	0239	,#ODB7	
2FC0 ODA7;	0240	,#ODA7	..224
2FC2 OD98;	0241	,#OD98	
2FC4 OD88;	0242	,#OD88	
2FC6 OD79;	0243	,#OD79	
2FC8 OD6A;	0244	,#OD6A	
2FCA OD5B;	0245	,#OD5B	
2FCC OD4D;	0246	,#OD4D	
2FCE OD3E;	0247	,#OD3E	
2FD0 OD2F;	0248	,#OD2F	
2FD2 OD21;	0249	,#OD21	



ASP FFT Program  
Program Code

2FD4 OD13;	0250	,#OD13	
2FD6 OD04;	0251	,#OD04	
2FD8 OCF6;	0252	,#OCF6	
2FDA OCE9;	0253	,#OCE9	
2FDC OCDB;	0254	,#OCDB	
2FDE OCCD;	0255	,#OCCD	
2FE0 OCBF;	0256	,#OCBF	..240
2FE2 OCB2;	0257	,#OCB2	
2FE4 OCA5;	0258	,#OCA5	
2FE6 OC97;	0259	,#OC97	
2FE8 OC8A;	0260	,#OC8A	
2FEA OC7D;	0261	,#OC7D	
2FEC OC70;	0262	,#OC70	
2FEE OC63;	0263	,#OC63	
2FF0 OC57;	0264	,#OC57	
2FF2 OC4A;	0265	,#OC4A	
2FF4 OC3D;	0266	,#OC3D	
2FF6 OC31;	0267	,#OC31	
2FF8 OC25;	0268	,#OC25	
2FFA OC18;	0269	,#OC18	
2FFC OC0C;	0270	,#OC0C	
2FFE OC00;	0271	,#OC00	..255
3000 ;	0272	..#####	
3000 ;	0273	..END OF FIX740.SR	



## APPENDIX B

### RUN TIME CODE

#### B.1 RUN TIME PROGRAMS

##### B.1.1 SYSRUN (System Run)

```
0000 ;          0001
0000 ;          0002 ..SYSRUN.SR          21 JAN 80          7:15 PM
0000 ;          0003

0000 ;          0241
0000 ;          0242 ..#####
0000 ;          0243 ..SYSRUN.SR          SYSTEM RUN
0000 ;          0244 ..
0000 ;          0245 ..THIS PROGRAM STARTS AT LOCATION #7000 WITH P=0
0000 ;          0246 ..IT SAVES THE SYSTEM REGISTERS
0000 ;          0247 ..INITIALIZES THE PROGRAM REGISTERS
0000 ;          0248 ..AND RUNS A SELECTED PROGRAM
0000 ;          0249 ..
0000 ;          0250 ..THEN MAY (BY SELECTION) EITHER
0000 ;          0251 .. 1) BREAK FROM USER PROGRAM
0000 ;          0252 ..     SAVE USER REGS
0000 ;          0253 ..     RETURN SYSTEM REGS
0000 ;          0254 ..     PRINT USER REGS
0000 ;          0255 ..     RETURN TO MONITOR
0000 ;          0256 ..OR 2) BREAK FROM USER PROGRAM
0000 ;          0257 ..     SAVE USER REGS
0000 ;          0258 ..     RETURN SYSTEM REGS
0000 ;          0259 ..     RETURN TO MONITOR
0000 ;          0260 ..OR 3) BREAK FROM USER PROGRAM
0000 ;          0261 ..     SAVE USER REGS
0000 ;          0262 ..     RETURN SYSTEM REGS
0000 ;          0263 ..     PRINT USER MEMORY
0000 ;          0264 ..     RETURN USER REGS
0000 ;          0265 ..     RETURN TO USER PROGRAM
0000 ;          0266 ..#####
```

ASP FFT Program  
Run Time Code

```

0000 ;          0267
0000 ;          0268
0000 ;          0269 ..SYSTEM REGISTER NAMES
0000 ;          0270
0000 ;          0271          PTER=#00; SUB=#03
0000 ;          0272          CINTER=#0D; AUX=#0E; CHAR=#0F
0000 ;          0273
0000 ;          0274
0000 ;          0275 ..EXTERNAL ROUTINES
0000 ;          0276
0000 ;          0277          START=#803F; DELAY1=#80EF; SYSLIN=#808D
0000 ;          0278          TYPE5D=#819C; TYPE2=#81AE; WRAM=#8C00
0000 ;          0279
0000 ;          0280
0000 ;          0281 ..VECTORS TO INTERNAL ROUTINES
0000 ;          0282          ORG #7000
7000 C07018;    0283          LBR SYSRUN          ..SYSTEM RUN
7003 C07073;    0284          LBR RGPRNT          ..REGISTER PRINT, QUIT
7006 C0707A;    0285          LBR PRGDUN          ..PROGRAM DONE SO QUIT
7009 ;          0286
7009 C070F6;    0287          LBR RGDATA          ..PRINT REGISTER DATA
700C C070FE;    0288          LBR FFTRAM          ..PRINT FFT FRAM
700F C07106;    0289          LBR PFTDTA          ..PRINT P-FFT DATA
7012 C0710E;    0290          LBR SFTDTA          ..PRINT S-FFT DATA
7015 C07116;    0291          LBR FFTDTA          ..PRINT FFT OUTPUT DATA
7018 ;          0292
7018 ;          0293
7018 ;          0294 ..BEGIN HERE
7018 ;          0295 SYSRUN:          ..SYSTEM RUN
7018 7100;      0296          DIS ,#00          ..DISABLE INTERRUPTS
701A ;          0297
701A ;          0298 ..SAVE SYSTEM RE.1
701A F872B6;    0299          LDI A.1(INTBL); PHI R6
701D F85EA6;    0300          LDI A.0(INTBL); PLO R6
7020 9E5616;    0301          GHI RE; STR R6; INC R6
7023 ;          0302
7023 ;          0303 ..SET PC TO LOAD PROGRAM REGISTERS
7023 F870B3;    0304          LDI A.1(LDPREG); PHI PC
7026 F82AA3;    0305          LDI A.0(LDPREG); PLO PC
7029 D3;        0306          SEP PC
702A ;          0307

```



702A ;	0308 LDPREG:	..LOAD PROGRAM REGISTERS
702A ;	0309 ..LOAD VARIABLE PROGRAM REGISTERS	
702A E6;	0310 SEX R6	
702B 72B072AO;	0311 LDXA ;PHI ZAP ;LDXA ;PLO ZAP	
702F 72B872A8;	0312 LDXA ;PHI FRP ;LDXA ;PLO FRP	
7033 72B972A9;	0313 LDXA ;PHI MP ;LDXA ;PLO MP	
7037 72BC72AC;	0314 LDXA ;PHI ZAP1;LDXA ;PLO ZAP1	
703B 72BA72AA;	0315 LDXA ;PHI ZAP2;LDXA ;PLO ZAP2	
703F 72BB72AB;	0316 LDXA ;PHI ZAP3;LDXA ;PLO ZAP3	
7043 72BD72AD;	0317 LDXA ;PHI MA ;LDXA ;PLO MA	
7047 72BE72AE;	0318 LDXA ;PHI MQ ;LDXA ;PLO MQ	
704B 72BF72AF;	0319 LDXA ;PHI AC ;LDXA ;PLO AC	
704F ;	0320	
704F ;	0321 ..NOW LOAD STATIC PROGRAM REGISTERS	
704F F800;	0322 LDI A.1(INTP)	
7051 B1;	0323 PHI R1	
7052 F850;	0324 LDI A.0(INTP)	
7054 A1;	0325 PLO R1	..SET R1 AS INTP PC
7055 F831;	0326 LDI A.1(STACK)	
7057 B2;	0327 PHI R2	
7058 F840;	0328 LDI A.0(STACK)	
705A A2;	0329 PLO R2	..SET STACK POINTER
705B F800;	0330 LDI A.1(CALL)	
705D B4;	0331 PHI R4	
705E F833;	0332 LDI A.0(CALL)	
7060 A4;	0333 PLO R4	..SET CALL ROUTINE PC
7061 F800;	0334 LDI A.1(RETN)	
7063 B5;	0335 PHI R5	
7064 F843;	0336 LDI A.0(RETN)	
7066 A5;	0337 PLO R5	..AND RETURN ROUTINE PC
7067 F831;	0338 LDI A.1(DSTACK)	
7069 B7;	0339 PHI R7	
706A F880;	0340 LDI A.0(DSTACK)	
706C A7;	0341 PLO R7	..SET DSTACK POINTER
706D ;	0342	
706D ;	0343 ..GET GOING IN PROGRAM	
706D D40000;	0344 CALL #0000	..CALL PROGRAM
7070 ;	0345	..(OR LONG BRANCH,
7070 ;	0346	..OR SEP TO ANOTHER REG)
7070 ;	0347	
7070 ;	0348 ..IF RETURN HERE (BY EXIT) QUIT TO MONITOR	
7070 C0707A;	0349 LBR PRGDUN	..GOTO PROGRAM DONE
7073 ;	0350	
7073 ;	0351	

ASP FFT Program  
Run Time Code

```

7073 ; 0352 ..*****
7073 ; 0353
7073 ; 0354 RGPRT: ..REGISTER PRINT
7073 ; 0355 ..ENTER HERE TO PRINT REGISTERS, THEN QUIT
7073 ; 0356
7073 ; 0357 ..CALL PRINT MEMORY
7073 ; 0358 ..WITH ADDRESS AND LENGTH OF REGISTER OUTPUT TABLE
7073 D4713D; 0359 CALL PRMEM
7076 70EE; 0360 ,A(OUTDIR)
7078 70F4; 0361 ,A(OUTDIR+6) ..TO POINT TO LENGTH
707A ; 0362
707A ; 0363 ..FALL THROUGH TO PROGRAM DONE
707A ; 0364
707A ; 0365
707A ; 0366 ..*****
707A ; 0367
707A ; 0368 PRGDUN: ..PROGRAM DONE
707A ; 0369 ..THIS SAVES REGISTERS, THEN QUIT (TO MONITOR)
707A ; 0370
707A ; 0371 ..SAVE PROGRAM REGS (USE R1 AS POINTER)
707A F873B1; 0372 LDI A.1(OUTTBL+31); PHI R1
707D F8FFA1; 0373 LDI A.0(OUTTBL+31); PLO R1
7080 E1; 0374 SEX R1
7081 ; 0375
7081 ; 0376 ..THEN STORE THEM
7081 8F739F73; 0377 GLO RF ;STXD ;GHI RF ;STXD
7085 8E739E73; 0378 GLO RE ;STXD ;GHI RE ;STXD
7089 8D739D73; 0379 GLO RD ;STXD ;GHI RD ;STXD
708D 8C739C73; 0380 GLO RC ;STXD ;GHI RC ;STXD
7091 8B739B73; 0381 GLO RB ;STXD ;GHI RB ;STXD
7095 8A739A73; 0382 GLO RA ;STXD ;GHI RA ;STXD
7099 89739973; 0383 GLO R9 ;STXD ;GHI R9 ;STXD
709D 88739873; 0384 GLO R8 ;STXD ;GHI R8 ;STXD
70A1 87739773; 0385 GLO R7 ;STXD ;GHI R7 ;STXD
70A5 86739673; 0386 GLO R6 ;STXD ;GHI R6 ;STXD
70A9 85739573; 0387 GLO R5 ;STXD ;GHI R5 ;STXD
70AD 84739473; 0388 GLO R4 ;STXD ;GHI R4 ;STXD
70B1 83739373; 0389 GLO R3 ;STXD ;GHI R3 ;STXD
70B5 82739273; 0390 GLO R2 ;STXD ;GHI R2 ;STXD
70B9 F8FF7373; 0391 LDI #FF ;STXD ;STXD
70BD 80739073; 0392 GLO R0 ;STXD ;GHI R0 ;STXD
70C1 ; 0393
70C1 ; 0394 ..ASSURE R3 IS PC
70C1 F870B1; 0395 LDI A.1(PC3); PHI R1
70C4 F8C8A1; 0396 LDI A.0(PC3); PLO R1
70C7 D1; 0397 SEP R1
70C8 F870B3; 0398 PC3: LDI A.1(SYSBCK); PHI PC
70CB F8CFA3; 0399 LDI A.0(SYSBCK); PLO PC
70CE D3; 0400 SEP PC
70CF ; 0401

```

```

70CF ;          0402 ..GET SYSTEM RE.1 BACK
70CF F872B1;    0403 SYSBCK: LDI A.1(INTBL); PHI R1
70D2 F85EAL;    0404         LDI A.0(INTBL); PLO R1
70D5 01BE;      0405         LDN R1; PHI RE
70D7 ;          0406
70D7 ;          0407 ..NOW COPY GOUT20-ENTER
70D7 ;          0408 ..(FROM SYSTEM PROGRAM)
70D7 E3;        0409         SEX PC
70D8 F880BC;    0410         LDI A.1(DELAY1); PHI RC
70DB F8EFAC;    0411         LDI A.0(DELAY1); PLO RC
70DE F88CB2;    0412         LDI A.1(WRAM); PHI R2
70E1 F800A2;    0413         LDI A.0(WRAM); PLO R2
70E4 F880B5;    0414         LDI A.1(START); PHI R5
70E7 F83FA5;    0415         LDI A.0(START); PLO R5
70EA 6101;      0416         OUT 1,#01         ..SELECT RCA I/O GROUP
70EC 7155;      0417         DIS,#55         ..P=X=5 TO GOTO START
70EE ;          0418         ..DISABLED
70EE ;          0419
70EE ;          0420
70EE ;          0421 OUTDIR:         ..OUTGOING TABLE DIRECTORY
70EE 73E0;      0422         ,A(OUTTBL)
70F0 7401;      0423         ,A(OUTTBL+33)
70F2 73E0;      0424         ,A(OUTTBL)
70F4 0020;      0425         ,#0020
70F6 ;          0426
70F6 ;          0427
70F6 ;          0428 ..*****
70F6 ;          0429
70F6 ;          0430 ..ENTRY POINTS FOR PRINTING MEMORY BLOCKS
70F6 ;          0431 ..REGISTER DATA, FFT FRAM, P-FFT OR S-FFT INPUT,
70F6 ;          0432 ..         OR FFT OUTPUT
70F6 ;          0433 ..
70F6 ;          0434 ..USER PROGRAM DOES A CALL #### TO GET HERE
70F6 ;          0435 ..THIS CALLS PRINT MEMORY
70F6 ;          0436 ..AND EXITS BACK TO USER PROGRAM
70F6 ;          0437 ..(IT MAY REQUIRE A HELPER SECTION TO MAKE UP
70F6 ;          0438 .. FOR LOST CODE)
70F6 ;          0439 ..
70F6 ;          0440 ..NOTE: P MUST BE 3 (PC)
70F6 ;          0441
70F6 ;          0442 RGDATA:         ..PRINT REGISTER DATA
70F6 ;          0443         ..P MUST BE 3 (PC)
70F6 D4713D;    0444         CALL PRMEM
70F9 70EE;      0445         ,A(OUTDIR)
70FB 70F4;      0446         ,A(OUTDIR+6)         ..TO POINT TO LENGTH
70FD D5;        0447         EXIT
70FE ;          0448

```



ASP FFT Program  
Run Time Code

70FE ;	0449 FFTRAM:	..PRINT FFT FRAM
70FE D4713D;	0450	CALL PRMEM
7101 712F;	0451	,A(FRMDIR)
7103 7135;	0452	,A(FRMDIR+6)
7105 D5;	0453	EXIT
7106 ;	0454	
7106 ;	0455 PFTDTA:	..PRINT P-FFT INPUT DATA
7106 D4713D;	0456	CALL PRMEM
7109 30A5;	0457	,A(YPTBL)
710B 30CC;	0458	,A(PFFTLN)
710D D5;	0459	EXIT
710E ;	0460	
710E ;	0461 SFTDTA:	..PRINT S-FFT INPUT DATA
710E D4713D;	0462	CALL PRMEM
7111 30AE;	0463	,A(YSTBL)
7113 30CE;	0464	,A(SFFTLN)
7115 D5;	0465	EXIT
7116 ;	0466	
7116 ;	0467 FFTDTA:	..PRINT FFT OUTPUT DATA
7116 ;	0468	..PUT BASE ADDRESS INTO FFT OUTPUT DIRECTORY
7116 F830B8;	0469	LDI A.1(BASE); PHI FRP
7119 F8D8A8;	0470	LDI A.0(BASE); PLO FRP
711C F871BB;	0471	LDI A.1(FFTDIR); PHI ZAP3
711F F837AB;	0472	LDI A.0(FFTDIR); PLO ZAP3
7122 485B1B;	0473	LDA FRP; STR ZAP3; INC ZAP3
7125 485B;	0474	LDA FRP; STR ZAP3
7127 ;	0475	
7127 ;	0476	..THEN CALL PRINT MEMORY
7127 D4713D;	0477	CALL PRMEM
712A 7137;	0478	,A(FFTDIR)
712C 30D0;	0479	,A(FFTLN)
712E D5;	0480	EXIT
712F ;	0481	
712F ;	0482 FRMDIR:	..FFT RAM DIRECTORY
712F 30A0;	0483	,#30A0
7131 3100;	0484	,A(FRAM+256)
7133 30A0;	0485	,#30A0
7135 0060;	0486	,#0060
7137 ;	0487	
7137 ;	0488 FFTDIR:	..FFT OUTPUT DIRECTORY
7137 0000;	0489	,#0000
7139 3E01;	0490	,A(YSLWA+1)
713B 3A01;	0491	,A(YSFWA)
713D ;	0492	
713D ;	0493	



```

713D ;      0494 ..*****
713D ;      0495
713D ;      0496 PRMEM:                ..PRINT MEMORY BLOCK
713D ;      0497 ..THIS IS CALLED TO PRINT A BLOCK OF MEMORY
713D ;      0498 ..R6 IS AT START ADDRESS POINTER
713D ;      0499 ..AT R6+2 IS LENGTH POINTER
713D ;      0500 ..IT HANDLES WRAPAROUND
713D ;      0501 ..
713D ;      0502 ..NOTE: P MUST BE 3 (PC)
713D ;      0503
713D ;      0504 ..DISABLE INTERRUPTS
713D E37133; 0505             SEX PC; DIS ,#33 ..IE=0, X=3, P=3
7140 ;      0506
7140 ;      0507 ..SAVE PROGRAM REGS (USE R1 AS POINTER)
7140 F873B1; 0508             LDI A.1(OUTTBL+31); PHI R1
7143 F8FFA1; 0509             LDI A.0(OUTTBL+31); PLO R1
7146 E1;     0510             SEX R1
7147 ;      0511
7147 ;      0512 ..THEN STORE THEM
7147 ;      0513 ..(STORE R6+4 FOR CORRECT RETURN POINT)
7147 8F739F73; 0514             GLO RF ;STXD ;GHI RF ;STXD
714B 8E739E73; 0515             GLO RE ;STXD ;GHI RE ;STXD
714F 8D739D73; 0516             GLO RD ;STXD ;GHI RD ;STXD
7153 8C739C73; 0517             GLO RC ;STXD ;GHI RC ;STXD
7157 8B739B73; 0518             GLO RB ;STXD ;GHI RB ;STXD
715B 8A739A73; 0519             GLO RA ;STXD ;GHI RA ;STXD
715F 89739973; 0520             GLO R9 ;STXD ;GHI R9 ;STXD
7163 88739873; 0521             GLO R8 ;STXD ;GHI R8 ;STXD
7167 87739773; 0522             GLO R7 ;STXD ;GHI R7 ;STXD
716B 86FC0473; 0523             GLO R6 ;ADI #04 ;STXD
716F 967C0073; 0524             GHI R6 ;ADCI #00 ;STXD
7173 85739573; 0525             GLO R5 ;STXD ;GHI R5 ;STXD
7177 84739473; 0526             GLO R4 ;STXD ;GHI R4 ;STXD
717B 83739373; 0527             GLO R3 ;STXD ;GHI R3 ;STXD
717F 82739273; 0528             GLO R2 ;STXD ;GHI R2 ;STXD
7183 F8FF7373; 0529             LDI #FF ;STXD ;STXD
7187 80739073; 0530             GLO R0 ;STXD ;GHI R0 ;STXD
718B ;      0531
718B ;      0532 ..MOVE @R6 -> R9 (DIRECTORY POINTER)
718B 46B9;    0533             LDA R6; PHI R9
718D 46A9;    0534             LDA R6; PLO R9
718F ;      0535
718F ;      0536 ..MOVE @R6 -> RA (LENGTH POINTER)
718F 46BA;    0537             LDA R6; PHI RA
7191 46AA;    0538             LDA R6; PLO RA
7193 ;      0539
7193 ;      0540 ..MOVE @R9 -> R(PTER) (START ADDRESS)
7193 49B0;    0541             LDA R9; PHI PTER
7195 49A0;    0542             LDA R9; PLO PTER
7197 ;      0543 ..LEAVE R9 AT LWA+1 VALUE
7197 ;      0544

```

ASP FFT Program  
Run Time Code

7197 ;	0545 ..MOVE @RA -> R(CNTER) (LENGTH VALUE)
7197 4ABD;	0546 LDA RA; PHI CNTER
7199 4AAD;	0547 LDA RA; PLO CNTER
719B ;	0548
719B ;	0549 ..GET SYSTEM RE.1 BACK
719B F872B1;	0550 LDI A.1(INTBL); PHI R1
719E F85EA1;	0551 LDI A.0(INTBL); PLO R1
71A1 01BE;	0552 LDN R1; PHI RE
71A3 ;	0553
71A3 ;	0554 ..NOW COPY GOUT20-ENTER
71A3 ;	0555 ..(FROM SYSTEM PROGRAM)
71A3 E3;	0556 SEX PC
71A4 F880BC;	0557 LDI A.1(DELAY1); PHI RC
71A7 F8EFAC;	0558 LDI A.0(DELAY1); PLO RC
71AA F88CB2;	0559 LDI A.1(WRAM); PHI R2
71AD F800A2;	0560 LDI A.0(WRAM); PLO R2
71B0 F871B5;	0561 LDI A.1(SUBPTR); PHI R5
71B3 F8BAA5;	0562 LDI A.0(SUBPTR); PLO R5
71B6 6101;	0563 OUT 1,#01 ..SELECT RCA I/O GROUP
71B8 7155;	0564 DIS,#55 ..P=X=5 TO GOTO SUB POINTER
71BA ;	0565 ..DISABLED
71BA ;	0566
71BA ;	0567 ..SET UP SUBROUTINE POINTER
71BA F881B3;	0568 SUBPTR: LDI A.1(TYPE5D); PHI SUB
71BD F89CA3;	0569 LDI A.0(TYPE5D); PLO SUB
71C0 ;	0570
71C0 ;	0571 ..TYPE CR/LF/LF
71C0 D30D;	0572 SEP SUB; ,#0D ..#0D = CR
71C2 D30A;	0573 SEP SUB; ,#0A ..#0A = LF
71C4 D30A;	0574 SEP SUB; ,#0A
71C6 ;	0575
71C6 ;	0576 LNOUT: ..LINE OUT
71C6 ;	0577 ..TYPE CARRIAGE RETURN & LINE FEED
71C6 D30D;	0578 SEP SUB; ,#0D ..#0D = CR
71C8 D30A;	0579 SEP SUB; ,#0A ..#0A = LF
71CA ;	0580
71CA ;	0581 ..OUTPUT HIGH ADDRESS
71CA 90BF;	0582 GHI PTR; PHI CHAR
71CC F8AEA3;	0583 LDI A.0(TYPE2); PLO SUB
71CF D3;	0584 SEP SUB
71D0 ;	0585
71D0 ;	0586 ..OUTPUT LOW ADDRESS
71D0 80BF;	0587 GLO PTR; PHI CHAR
71D2 F8AEA3;	0588 LDI A.0(TYPE2); PLO SUB
71D5 D3;	0589 SEP SUB
71D6 ;	0590
71D6 ;	0591 TSPACE: ..TYPE SPACE
71D6 D320;	0592 SEP SUB; ,#20 ..#20 = SPACE
71D8 ;	0593

```

71D8 ;          0594 TLOOP:                ..TYPE LOOP
71D8 ;          0595 ..FETCH ONE BYTE FOR TYPING
71D8 40BF;      0596          LDA PTER; PHI CHAR
71DA F8AEA3;    0597          LDI A.0(TYPE2); PLO SUB
71DD D3;        0598          SEP SUB                ..TYPE 2 HEX DIGITS
71DE ;          0599
71DE ;          0600 ..CHECK FOR DONE
71DE 2D;        0601          DEC COUNTER
71DF 8DCA71E7;  0602          GLO COUNTER; LBNZ LWACHK
71E3 9DC2720C;  0603          GHI COUNTER; LBZ EXPRMM
71E7 ;          0604
71E7 ;          0605 LWACHK:                ..FIFO LWA+1 CHECK
71E7 E9;        0606          SEX R9
71E8 1980F7;    0607          INC R9; GLO PTER; SM
71EB 299077;    0608          DEC R9; GHI PTER; SMB
71EE CB71FA;    0609          LBNF LNCHK                ..IF PTER < LWA+1
71F1 ;          0610          ..THEN GOTO LINE CHECK
71F1 ;          0611
71F1 ;          0612 ..ELSE PTER => LWA+1, RESET TO FWA
71F1 1919;      0613          INC R9; INC R9
71F3 49B0;      0614          LDA R9; PHI PTER
71F5 09A0;      0615          LDN R9; PLO PTER
71F7 292929;    0616          DEC R9; DEC R9; DEC R9
71FA ;          0617
71FA ;          0618 LNCHK:                ..LINE CHECK
71FA ;          0619 ..SEE IF R(COUNTER) DIV BY 16
71FA 8DFA0F;    0620          GLO COUNTER; ANI #0F ..CHECK #X0
71FD CA7205;    0621          LBNZ WRDCHK
7200 D33B;      0622          SEP SUB; ,#3B                ..#3B = "; "
7202 C071C6;    0623          LBR LNOUT
7205 ;          0624
7205 ;          0625 WRDCHK:                ..WORD CHECK
7205 ;          0626 ..SEE IF R(PTER) DIV BY 2
7205 F6;        0627          SHR                ..CHECK LOW BIT
7206 C371D8;    0628          LBDF TLOOP                ..IF NOT 0, OUT NEXT BYTE
7209 C071D6;    0629          LBR TSPACE                ..ELSE 1, OUT SPACE
720C ;          0630
720C ;          0631

```

ASP FFT Program  
Run Time Code

720C ;	0632 ..-----
720C ;	0633 EXPRMM: ..EXIT PRINT MEMORY
720C ;	0634 ..RESTORE PROGRAM REGS, THEN SKIP BACK TO USER PRGM
720C ;	0635
720C ;	0636 ..GET R3 AS PC
720C F872B3;	0637 LDI A.1(PCIS3); PHI PC
720F F813A3;	0638 LDI A.0(PCIS3); PLO PC
7212 D3;	0639 SEP PC
7213 ;	0640
7213 ;	0641 PCIS3: ..PC IS 3
7213 ;	0642 ..RESTORE PROGRAM REGS
7213 ;	0643 ..GET R1 TO OUTTBL
7213 F873B1;	0644 LDI A.1(OUTTBL); PHI R1
7216 F8EOA1;	0645 LDI A.0(OUTTBL); PLO R1
7219 ;	0646
7219 ;	0647 ..RESTORE THEM
7219 41B041A0;	0648 LDA R1; PHI R0; LDA R1; PLO R0
721D 4141;	0649 LDA R1; LDA R1
721F 41B241A2;	0650 LDA R1; PHI R2; LDA R1; PLO R2
7223 4141;	0651 LDA R1; LDA R1
7225 41B441A4;	0652 LDA R1; PHI R4; LDA R1; PLO R4
7229 41B541A5;	0653 LDA R1; PHI R5; LDA R1; PLO R5
722D 41B641A6;	0654 LDA R1; PHI R6; LDA R1; PLO R6
7231 41B741A7;	0655 LDA R1; PHI R7; LDA R1; PLO R7
7235 41B841A8;	0656 LDA R1; PHI R8; LDA R1; PLO R8
7239 41B941A9;	0657 LDA R1; PHI R9; LDA R1; PLO R9
723D 41BA41AA;	0658 LDA R1; PHI RA; LDA R1; PLO RA
7241 41BB41AB;	0659 LDA R1; PHI RB; LDA R1; PLO RB
7245 41BC41AC;	0660 LDA R1; PHI RC; LDA R1; PLO RC
7249 41BD41AD;	0661 LDA R1; PHI RD; LDA R1; PLO RD
724D 41BE41AE;	0662 LDA R1; PHI RE; LDA R1; PLO RE
7251 41BF41AF;	0663 LDA R1; PHI RF; LDA R1; PLO RF
7255 ;	0664
7255 ;	0665 ..RESTORE R1
7255 F800B1;	0666 LDI A.1(INTP); PHI R1
7258 F850A1;	0667 LDI A.0(INTP); PLO R1
725B ;	0668
725B ;	0669 ..ENABLE INTERRUPTS AND RETURN TO USER PROGRAM
725B E37035;	0670 SEX PC; RET ,#35 ..IE=0, X=3, P=5
725E ;	0671



```

725E ;      0672 ..*****
725E ;      0673 ..TABLES
725E ;      0674
725E ;      0675 INTBL:          ..INGOING REG TABLE
725E 00;      0676 ,#00      ..SYSTEM RE.1
725F 0000;    0677 ,#0000    ..PROGRAM ZAP
7261 3000;    0678 ,#3000    ..PROGRAM FRP (SET TO FRAM)
7263 0000;    0679 ,#0000    ..PROGRAM MP
7265 0000;    0680 ,#0000    ..PROGRAM ZAP1
7267 0000;    0681 ,#0000    ..PROGRAM ZAP2
7269 0000;    0682 ,#0000    ..PROGRAM ZAP3
726B 0000;    0683 ,#0000    ..PROGRAM MA
726D 0000;    0684 ,#0000    ..PROGRAM MQ
726F 0000;    0685 ,#0000    ..PROGRAM AC
7271 ;      0686
7271 ;      0687          ORG #73E0
73E0 ;      0688 OUTTBL:      ..OUTCOMING REG TABLE
73E0 0000;    0689 ,#0000    ..PROGRAM R0
73E2 0000;    0690 ,#0000    ..PROGRAM R1
73E4 0000;    0691 ,#0000    ..PROGRAM R2
73E6 0000;    0692 ,#0000    ..PROGRAM R3
73E8 0000;    0693 ,#0000    ..PROGRAM R4
73EA 0000;    0694 ,#0000    ..PROGRAM R5
73EC 0000;    0695 ,#0000    ..PROGRAM R6
73EE 0000;    0696 ,#0000    ..PROGRAM R7
73F0 0000;    0697 ,#0000    ..PROGRAM R8
73F2 0000;    0698 ,#0000    ..PROGRAM R9
73F4 0000;    0699 ,#0000    ..PROGRAM RA
73F6 0000;    0700 ,#0000    ..PROGRAM RB
73F8 0000;    0701 ,#0000    ..PROGRAM RC
73FA 0000;    0702 ,#0000    ..PROGRAM RD
73FC 0000;    0703 ,#0000    ..PROGRAM RE
73FE 0000;    0704 ,#0000    ..PROGRAM RF
7400 ;      0705
7400 ;      0706
7400 ;      0707 DONE:      END

```

ASP FFT Program  
Run Time Code

B.1.2 LDFIFO (Load FIFO)

```

0000 ;          0001
0000 ;          0002 ..LDFIFO.SR          21 OCT 79          4:00 PM
0000 ;          0003

0000 ;          0101
0000 ;          0102 ..#####
0000 ;          0103 ..LDFIFO.SR          LOAD FIFO
0000 ;          0104 ..
0000 ;          0105 ..THERE ARE 4 FORMS TO THIS ROUTINE
0000 ;          0106 ..          1) ZERO YPFIFO,
0000 ;          0107 ..          2) ZERO YSFIFO,
0000 ;          0108 ..          3) MANUAL INPUT, AND
0000 ;          0109 ..          4) MODIFIED SINE
0000 ;          0110 ..AS DESCRIBED BELOW
0000 ;          0111 ..#####
0000 ;          0112
0000 ;          0113
0000 ;          0114 ..EXTERNAL ROUTINES
0000 ;          0115
0000 ;          0116          GOUT20=#83F9          ..VECTOR TO ENTER UTILITY
0000 ;          0117
0000 ;          0118
0000 ;          0119 ..VECTORS TO INTERNAL ROUTINES
0000 ;          0120
0000 ;          0121          ORG #5100
5100 C0510C;      0122          LBR YPZERO          ..ZERO YPFIFO
5103 C0512F;      0123          LBR YSZERO          ..ZERO YSFIFO
5106 C05152;      0124          LBR MANINP          ..MANUAL INPUT
5109 C051D5;      0125          LBR MODSIN          ..MODIFIED SINE
510C ;          0126
510C ;          0127
510C ;          0128 ..BEGIN HERE
510C ;          0129
510C ;          0130

```

```

510C ;          0131 ..#####
510C ;          0132 ..YPZERO.SR              ZERO YPFIFO
510C ;          0133 ..#####
510C ;          0134
510C ;          0135
510C 7100;      0136 YPZERO: DIS ,#00          ..INITIALIZE
510E F851;      0137             LDI A.1(YPSTRT)
5110 B3;        0138             PHI PC
5111 F815;      0139             LDI A.0(YPSTRT)
5113 A3;        0140             PLO PC
5114 D3;        0141             SEP PC
5115 ;          0142
5115 F83A;      0143 YPSTRT: LDI A.1(YPLWA)
5117 BB;        0144             PHI ZAP3
5118 F800;      0145             LDI A.0(YPLWA)
511A AB;        0146             PLO ZAP3          ..R(ZAP3) = LAST ADDRESS
511B EB;        0147             SEX ZAP3
511C ;          0148
511C ;          0149 PTZROP:          ..PUT ZERO INTO YPFIFO
511C F800;      0150             LDI #00
511E 73;        0151             STXD
511F 73;        0152             STXD
5120 ;          0153
5120 ;          0154 DNPZRO:          ..FIRST ADDRESS CHECK
5120 8B;        0155             GLO ZAP3
5121 FD00;      0156             SDI A.0(YPFWA-1)
5123 CB511C;    0157             LBNF PTZROP
5126 9B;        0158             GHI ZAP3
5127 7D36;      0159             SDBI A.1(YPFWA-1)
5129 CB511C;    0160             LBNF PTZROP
512C ;          0161
512C C083F9;    0162             LBR GOUT20          ..JUMP OUT IF DONE
512F ;          0163
512F ;          0164
512F ;          0165

```

ASP FFT Program  
Run Time Code

```

512F ; 0166 ..#####
512F ; 0167 ..YSZERO.SR ZERO YSFIFO
512F ; 0168 ..#####
512F ; 0169
512F ; 0170
512F 7100; 0171 YSZERO: DIS ,#00 ..INITIALIZE
5131 F851; 0172 LDI A.1(YSSTRT)
5133 B3; 0173 PHI PC
5134 F838; 0174 LDI A.0(YSSTRT)
5136 A3; 0175 PLO PC
5137 D3; 0176 SEP PC
5138 ; 0177
5138 F83E; 0178 YSSTRT: LDI A.1(YSLWA)
513A BB; 0179 PHI ZAP3
513B F800; 0180 LDI A.0(YSLWA)
513D AB; 0181 PLO ZAP3 ..R(ZAP3) = LAST ADDRESS
513E EB; 0182 SEX ZAP3
513F ; 0183
513F ; 0184 PTZROS: ..PUT ZERO INTO YSFIFO
513F F800; 0185 LDI #00
5141 73; 0186 STXD
5142 73; 0187 STXD
5143 ; 0188
5143 ; 0189 DNSZRO: ..FIRST ADDRESS CHECK
5143 8B; 0190 GLO ZAP3
5144 FD00; 0191 SDI A.0(YSFWA-1)
5146 CB513F; 0192 LBNF PTZROS
5149 9B; 0193 GHI ZAP3
514A 7D3A; 0194 SDBI A.1(YSFWA-1)
514C CB513F; 0195 LBNF PTZROS
514F ; 0196
514F C083F9; 0197 LBR GOUT20 ..JUMP OUT IF DONE
5152 ; 0198
5152 ; 0199

```



```

5152 ;      0200 ..#####
5152 ;      0201 ..MANINP.SR                      ..MANUAL INPUT
5152 ;      0202 ..
5152 ;      0203 ..TO RUN THIS, ENTER INTO TABLE1
5152 ;      0204 ..      1. START ADDRESS
5152 ;      0205 ..      2. A. LWA+1, FWA POINTER LOCATION
5152 ;      0206 ..          EITHER ,A(PTBL1) FOR YPFIFO
5152 ;      0207 ..          OR ,A(STBL1) FOR YSFIFO
5152 ;      0208 ..      B. INITIAL VALUE
5152 ;      0209 ..      C. LENGTH OF BLOCK * 2
5152 ;      0210 ..          (#0400=512, #0020=16, ETC.)
5152 ;      0211 ..      D. INCREMENT/DECREMENT VALUE
5152 ;      0212 ..          (MAY BE #8000 TO #7FFF, INCL #0000)
5152 ;      0213 ..THEN
5152 ;      0214 ..      3. A. NEXT BLOCK'S INITIAL VALUE
5152 ;      0215 ..          (OR NOP PROGRAM TO USE OLD VALUE)
5152 ;      0216 ..      B. NEXT BLOCK'S LENGTH
5152 ;      0217 ..      C. NEXT BLOCK'S INC/DEC VALUE
5152 ;      0218 ..      ETC.
5152 ;      0219 ..OR
5152 ;      0220 ..      4. ENTER #ABXX TO STOP
5152 ;      0221 ..          (WHERE XX ARE DON'T CARES)
5152 ;      0222 ..#####
5152 ;      0223
5152 ;      0224
5152 ;      0225 ..ENTER HERE
5152 ;      0226 MANINP:                      ..MANUAL INPUT
5152 7100;      0227      DIS ,#00          ..INITIALIZE
5154 F851;      0228      LDI A.1(MNSTRT)
5156 B3;      0229      PHI PC
5157 F85B;      0230      LDI A.0(MNSTRT)
5159 A3;      0231      PLO PC
515A D3;      0232      SEP PC
515B ;      0233
515B ;      0234 ..GET R2 TO TABLE1
515B ;      0235 MNSTRT:                      ..MANUAL START
515B F852;      0236      LDI A.1(TABLE1)
515D B2;      0237      PHI R2
515E F832;      0238      LDI A.0(TABLE1)
5160 A2;      0239      PLO R2
5161 ;      0240
5161 ;      0241 ..LOAD START ADDRESS INTO R(MA)
5161 E2;      0242      SEX R2
5162 72;      0243      LDXA
5163 BD;      0244      PHI MA
5164 72;      0245      LDXA
5165 AD;      0246      PLO MA
5166 ;      0247

```

ASP FFT Program  
Run Time Code

5166 ;	0248	..LOAD LWA+1, FWA POINTER INTO R(ZAP1)
5166 72;	0249	LDXA
5167 BC;	0250	PHI ZAP1
5168 72;	0251	LDXA
5169 AC;	0252	PLO ZAP1
516A ;	0253	
516A ;	0254	..LOAD INITIAL VALUE INTO R(AC)
516A 72;	0255	LDXA
516B BF;	0256	PHI AC
516C 72;	0257	LDXA
516D AF;	0258	PLO AC
516E ;	0259	..LEAVE R2 AT BLOCK LENGTH
516E ;	0260	
516E ;	0261	..GET FREE LENGTH INTO R(ZAP2)
516E ;	0262	..(FREE LENGTH = BOUNDARY - START ADDRESS
516E EC;	0263	INFRLN: SEX ZAP1
516F 1C;	0264	INC ZAP1
5170 8D;	0265	GLO MA
5171 F5;	0266	SD
5172 AA;	0267	PLO ZAP2
5173 2C;	0268	DEC ZAP1
5174 9D;	0269	GHI MA
5175 75;	0270	SDB
5176 BA;	0271	PHI ZAP2
5177 ;	0272	
5177 ;	0273	..CHECK FOR BOUNDARY OK
5177 E2;	0274	SEX R2
5178 12;	0275	INC R2
5179 8A;	0276	GLO ZAP2
517A F5;	0277	SD
517B 22;	0278	DEC R2
517C 9A;	0279	GHI ZAP2
517D 75;	0280	SDB
517E C35187;	0281	LBDF BDRYBD ..IF BLOCK > FREE
5181 ;	0282	..BOUNDARY BAD
5181 ;	0283	
5181 ;	0284	..BOUNDARY GOOD, STORE TRUE (#FF) INTO R(ZAP3.H)
5181 F8FF;	0285	LDI #FF
5183 BB;	0286	PHI ZAP3
5184 C0518A;	0287	LBR INPBLN
5187 ;	0288	
5187 ;	0289	BDRYBD: ..BOUNDARY BAD
5187 ;	0290	..STORE FALSE (#00) INTO R(ZAP3.H)
5187 F800;	0291	LDI #00
5189 BB;	0292	PHI ZAP3
518A ;	0293	

518A ;	0294 INPBLN:	..INPUT BLOCK LENGTH
518A ;	0295 ..GET BLOCK LENGTH INTO R(MP)	
518A 72;	0296 LDXA	
518B B9;	0297 PHI MP	
518C 72;	0298 LDXA	
518D A9;	0299 PLO MP	
518E ;	0300 ..LEAVE R2 AT INC/DEC VALUE	
518E ;	0301	
518E ;	0302 STRVLU:	..STORE VALUE
518E ;	0303 ..STORE R(AC) AT M(R(MA))	
518E 9F;	0304 GHI AC	
518F 5D;	0305 STR MA	
5190 1D;	0306 INC MA	
5191 8F;	0307 GLO AC	
5192 5D;	0308 STR MA	
5193 1D;	0309 INC MA	
5194 ;	0310	
5194 ;	0311 ..INC/DEC R(AC)	
5194 E2;	0312 SEX R2	
5195 12;	0313 INC R2	
5196 8F;	0314 GLO AC	
5197 F4;	0315 ADD	
5198 AF;	0316 PLO AC	
5199 22;	0317 DEC R2	
519A 9F;	0318 GHI AC	
519B 74;	0319 ADC	
519C BF;	0320 PHI AC	
519D ;	0321	
519D ;	0322 ..CHECK FOR WRAPAROUND	
519D ;	0323 ..CHECK BOUNDARY OK FLAG, IF TRUE THEN SKIP	
519D 9B;	0324 GHI ZAP3	
519E CA51B8;	0325 LBNZ BLKCHK	..TO BLOCK CHECK
51A1 ;	0326	
51A1 ;	0327 ..ELSE DECREMENT FREE LENGTH, CHECK FOR #00	
51A1 2A;	0328 DEC ZAP2	
51A2 2A;	0329 DEC ZAP2	
51A3 8A;	0330 GLO ZAP2	
51A4 CA51B8;	0331 LBNZ BLKCHK	..TO BLOCK CHECK
51A7 9A;	0332 GHI ZAP2	
51A8 CA51B8;	0333 LBNZ BLKCHK	..TO BLOCK CHECK
51AB ;	0334	
51AB ;	0335 ..DOWN TO #00, SET R(MA) TO FIFO FWA	
51AB 1C;	0336 INC ZAP1	
51AC 1C;	0337 INC ZAP1	
51AD EC;	0338 SEX ZAP1	
51AE 72;	0339 LDXA	
51AF BD;	0340 PHI MA	
51B0 F0;	0341 LDX	
51B1 AD;	0342 PLO MA	
51B2 ;	0343	

ASP FFT Program  
Run Time Code

51B2 ;	0344	..AND RETURN R(ZAP1) TO FIFO LWA+1	
51B2 2C;	0345	DEC ZAP1	
51B3 2C;	0346	DEC ZAP1	
51B4 2C;	0347	DEC ZAP1	
51B5 ;	0348		
51B5 ;	0349	..SET BOUNDARY OK FLAG TRUE	
51B5 F8FF;	0350	LDI #FF	
51B7 BB;	0351	PHI ZAP3	
51B8 ;	0352		
51B8 ;	0353	BLKCHK:	..BLOCK CHECK
51B8 ;	0354	..DECREMENT BLOCK LENGTH COUNT, CHECK FOR ZERO	
51B8 29;	0355	DEC MP	
51B9 29;	0356	DEC MP	
51BA 89;	0357	GLO MP	
51BB CA518E;	0358	LBNZ STRVLU	..TO STORE VALUE
51BE 99;	0359	GHI MP	
51BF CA518E;	0360	LBNZ STRVLU	..TO STORE VALUE
51C2 ;	0361		
51C2 ;	0362	..ELSE, NEW BLOCK	
51C2 ;	0363	..GET R2 TO NEXT INPUT VALUE	
51C2 12;	0364	INC R2	
51C3 12;	0365	INC R2	
51C4 ;	0366		
51C4 ;	0367	..IF #AB, THEN DONE	
51C4 E2;	0368	SEX R2	
51C5 FO;	0369	LDX	
51C6 FFAB;	0370	SMI #AB	
51C8 C251D2;	0371	LBZ END1	..TO END
51CB ;	0372		
51CB ;	0373	..ELSE, PUT INTO R(AC)	
51CB 72;	0374	LDXA	
51CC BF;	0375	PHI AC	..NOP (#C4) THIS
51CD 72;	0376	LDXA	
51CE AF;	0377	PLO AC	..AND THIS FOR OLD VALUE
51CF ;	0378		
51CF ;	0379	..GOTO INPUT NEW FREE LENGTH	
51CF C0516E;	0380	LBR INFRLN	
51D2 ;	0381		
51D2 ;	0382	END1:	..END OF MANUAL INPUT
51D2 C083F9;	0383	LBR GOUT20	
51D5 ;	0384		
51D5 ;	0385		



```

51D5 ; 0386 ..#####
51D5 ; 0387 ..MODSIN.SR                      MODIFIED SINE WAVE
51D5 ; 0388 ..
51D5 ; 0389 ..THIS PROGRAM MOVES THE SINE WAVE AT #4000 - #41FE
51D5 ; 0390 ..TO A CHOSEN OUTPUT FIFO WITH A SELECTED
51D5 ; 0391 ..FREQUENCY SCALING
51D5 ; 0392 ..(VIA SKIPPING TRANSFERRED VALUES)
51D5 ; 0393 ..
51D5 ; 0394 ..ENTER INTO TABLE1
51D5 ; 0395 ..FWA, LWA+1 POINTER, EITHER ,A(YPTBL2) FOR YPFIFO
51D5 ; 0396 ..                                OR ,A(YSTBL2) FOR YSFIFO
51D5 ; 0397 ..SKIP VALUE * 2 (#0002, #0004, #0006, ...)
51D5 ; 0398 ..#####
51D5 ; 0399
51D5 ; 0400
51D5 ; 0401 ..ENTER HERE
51D5 ; 0402 MODSIN:                      ..MODIFIED SINE WAVE
51D5 7100; 0403         DIS ,#00                      ..INITIALIZE
51D7 F851; 0404         LDI A.1(MSSTR)
51D9 B3; 0405         PHI PC
51DA F8DE; 0406         LDI A.0(MSSTR)
51DC A3; 0407         PLO PC
51DD D3; 0408         SEP PC
51DE ; 0409
51DE ; 0410 ..GET R2 TO TABLE2
51DE ; 0411 MSSTR:                      ..MANUAL SINE START
51DE F852; 0412         LDI A.1(TABLE2)
51E0 B2; 0413         PHI R2
51E1 F826; 0414         LDI A.0(TABLE2)
51E3 A2; 0415         PLO R2
51E4 ; 0416
51E4 ; 0417 ..LOAD FIFO POINTER INTO R(ZAP1)
51E4 E2; 0418         SEX R2
51E5 72; 0419         LDXA
51E6 BC; 0420         PHI ZAP1
51E7 72; 0421         LDXA
51E8 AC; 0422         PLO ZAP1
51E9 ; 0423 ..LEAVE R2 AT INCREMENT VALUE
51E9 ; 0424
51E9 ; 0425 ..LOAD FIFO FWA INTO R(MA)
51E9 EG; 0426         SEX ZAP1
51EA 72; 0427         LDXA
51EB BD; 0428         PHI MA
51EC 72; 0429         LDXA
51ED AD; 0430         PLO MA
51EE ; 0431 ..AND LEAVE R(ZAP1) AT FIFO LWA+1
51EE ; 0432

```

ASP FFT Program  
Run Time Code

51EE ;	0433	..GET R(MP) TO SINE FWA	
51EE F840;	0434	LDI A.1(SINFWA)	
51F0 B9;	0435	PHI MP	
51F1 F800;	0436	LDI A.0(SINFWA)	
51F3 A9;	0437	PLO MP	
51F4 ;	0438		
51F4 ;	0439	..PASS FROM M(R(MP)) TO M(R(MA))	
51F4 E9;	0440	PASS: SEX MP	
51F5 72;	0441	LDXA	
51F6 5D;	0442	STR MA	
51F7 1D;	0443	INC MA	
51F8 F0;	0444	LDX	
51F9 5D;	0445	STR MA	
51FA 1D;	0446	INC MA	
51FB 29;	0447	DEC MP	
51FC ;	0448	..AND LEAVE R(MP) WHERE IT WAS	
51FC ;	0449		
51FC ;	0450	..CHECK FOR FIFO POINTER TO END	
51FC EC;	0451	SEX ZAP1	
51FD 1C;	0452	INC ZAP1	
51FE 8D;	0453	GLO MA	
51FF F7;	0454	SM	
5200 2C;	0455	DEC ZAP1	
5201 9D;	0456	GHI MA	
5202 77;	0457	SMB	
5203 C3521B;	0458	LBDF END2	..IF PTR > BDRY, DONE
5206 ;	0459		
5206 ;	0460	..ELSE, INCREMENT SINE POINTER	
5206 E2;	0461	SEX R2	
5207 12;	0462	INC R2	
5208 89;	0463	GLO MP	
5209 F4;	0464	ADD	
520A A9;	0465	PLO MP	
520B 22;	0466	DEC R2	
520C 99;	0467	GHI MP	
520D 74;	0468	ADC	
520E B9;	0469	PHI MP	
520F ;	0470		
520F ;	0471	..CHECK FOR SINE POINTER OFF END	
520F 99;	0472	GHI MP	
5210 FF42;	0473	SMI A.1(SINLWA+2)	
5212 ;	0474		
5212 ;	0475	..IF POINTER < BOUNDARY, THEN PASS AGAIN	
5212 CB51F4;	0476	LBNE PASS	
5215 ;	0477		
5215 ;	0478	..ELSE, RESET SINE POINTER	
5215 F840;	0479	LDI A.1(SINFWA)	
5217 B9;	0480	PHI MP	
5218 ;	0481		

```

5218 ;                0482 ..THEN PASS AGAIN
5218 C051F4;          0483                LBR PASS
521B ;                0484
521B ;                0485 END2:                ..END OF MODIFIED SINE
521B C083F9;          0486                LBR GOUT20
521E ;                0487
521E ;                0488
521E ;                0489 ..#####
521E ;                0490 ..HERE ARE THE TABLES
521E ;                0491
521E 3601;            0492 PTBL2: ,A(YPFWA)
5220 3A01;            0493                ,A(YPLWA+1)
5222 3A01;            0494 STBL2: ,A(YSFWA)
5224 3E01;            0495                ,A(YSLWA+1)
5226 0000;            0496 TABLE2: ,#0000                ..FIFO POINTER
5228 0000;            0497                ,#0000                ..SKIP VALUE
522A ;                0498
522A ;                0499 ..*****
522A ;                0500
522A 3A01;            0501 PTBL1: ,A(YPLWA+1)
522C 3601;            0502                ,A(YPFWA)
522E 3E01;            0503 STBL1: ,A(YSLWA+1)
5230 3A01;            0504                ,A(YSFWA)
5232 0000;            0505 TABLE1: ,#0000                ..START ADDRESS
5234 0000;            0506                ,#0000                ..LWA+1, FWA POINTER
5236 0000;            0507                ,#0000                ..INITIAL VALUE
5238 0000;            0508                ,#0000                ..BLOCK LENGTH
523A 0000;            0509                ,#0000                ..INC/DEC VALUE
523C ;                0510
523C 0000;            0511                ,#0000                ..NEXT INITIAL VALUE
523E 0000;            0512                ,#0000                ..NEXT BLOCK LENGTH
5240 0000;            0513                ,#0000                ..NEXT INC/DEC VALUE
5242 ;                0514                ..ETC.
5242 ;                0515
5242 0000;            0516                ,#0000                ..UNTIL #ABXX
5244 ;                0517
5244 ;                0518
5244 ;                0519 ..#####
5244 ;                0520 ..END OF LDFIFO

```

ASP FFT Program  
Run Time Code

B.1.3 FFT COMPLETE TRACE

```

0000 ;          0001
0000 ;          0002 ..FFT COMPLETE TRACE.SR   5 MARCH 80       2:10 PM
0000 ;          0003
0000 ;          0004          ZAP=#00; PC=#03; FRP=#08; MP=#09
0000 ;          0005          ZAP2=#0A; ZAP3=#0B; ZAP1=#0C
0000 ;          0006          MA=#0D; MQ=#0E; AC=#0F
0000 ;          0007
0000 ;          0008 ..*****
0000 ;          0009 ..SET P-LENGTH = 64, S-LENGTH = 128
0000 ;          0010          ORG #3046
3046 11110000; 0011          ,#11110000          ..SET SLAST NOT #0000
304A ;          0012          ..          DT=#0000
304A ;          0013
304A ;          0014 ..SET GAIN = 1          ..(MAY BE CHANGED LATER)
304A ;          0015          ORG #3062
3062 0001;     0016          ,#0001
3064 ;          0017
3064 ;          0018 ..LDFIFO SET
3064 ;          0019          ORG #51D2
51D2 C0701A;   0020          ,#C0701A          ..$P5152 FOR FIFO INPUT
51D5 ;          0021          ORG #521B
521B C0701A;   0022          ,#C0701A          ..$P51D5 FOR SINE
521E ;          0023          ORG #5232
5232 3601522A; 0024          ,#3601; ,#522A          ..(#3A01,522E FOR S-FFT)
5236 7FFF00040000; 0025          ,#7FFF; ,#0004; ,#0000
523C 000003000000; 0026          ,#0000; ,#0300; ,#0000
5242 ABAB;     0027          ,#ABAB
5244 ;          0028          ORG #5226
5226 521E0020; 0029          ,#521E; ,#0020          ..(#5222 FOR S-FFT)
522A ;          0030
522A ;          0031 ..AFTER SYSRUN INITIALIZE, GOTO WORKER
522A ;          0032 ..(MAY BE CHANGED LATER)
522A ;          0033          ORG #706D
706D C00600;   0034          LBR #0600          ..CALL WORKER
7070 ;          0035          ..(#D41700 TO RUN P-FFT)
7070 ;          0036          ..(#D41703 TO RUN S-FFT)
7070 ;          0037

```



```

7070 ;          0038 ..*****
7070 ;          0039 ..EXTERNAL SUBROUTINES
7070 ;          0040          RGDATA=#7009          ..PRINT REGISTER DATA
7070 ;          0041          PFTDTA=#700F          ..PRINT P-FFT INPUT DATA
7070 ;          0042          SFTDTA=#7012          ..PRINT S-FFT INPUT DATA
7070 ;          0043          FFTDTA=#7015          ..PRINT FFT OUTPUT DATA
7070 ;          0044                                  ..(OR INTERMEDIATE RESULTS)
7070 ;          0045          PRMEM=#713D          ..PRINT MEMORY
7070 ;          0046
7070 ;          0047 ..SWITCH FFTDTA PRINT TO INCLUDE TOTLSH
7070 ;          0048 ..(DOES KILL "FFTRAM" PRINT, BUT THAT'S OK)
7070 ;          0049          ORG #712E
712E D4713D74507456;0050          CALL PRMEM; ,#7450; ,#7456
7135 D5;          0051          EXIT
7136 ;          0052
7136 ;          0053 ..MAKE A FFT OUTPUT PRINT (TO BE "CALL"ED)
7136 ;          0054          ORG #7500
7500 D4713D74007406;0055          CALL PRMEM; ,#7400; ,#7406
7507 D5;          0056          EXIT
7508 ;          0057
7508 ;          0058 ..-----
7508 ;          0059 ..ROUTINE TO BE CALLED FOR INPUT DATA FIFO
7508 ;          0060
7508 ;          0061          ORG #7600
7600 ;          0062 INPRNT:          ..INPUT PRINT
7600 ;          0063 ..CHECK FOR P-FFT
7600 F830BD;          0064          LDI #30; PHI RD
7603 F8C4AD;          0065          LDI #C4; PLO RD
7606 OD320E;          0066          LDN RD; BZ ISSFFT ..IF PFTFLG=#00, THEN
7609 ;          0067                                  ..GOTO PRINT S-FFT DATA
7609 ;          0068
7609 ;          0069 ..ELSE PRINT P-FFT DATA
7609 D4700F;          0070          CALL PFTDTA
760C 3011;          0071          BR JMPBCK          ..THEN JUMP BACK
760E ;          0072
760E ;          0073 ISSFFT:          ..IS S-FFT
760E ;          0074 ..PRINT S-FFT DATA
760E D47012;          0075          CALL SFTDTA
7611 ;          0076
7611 ;          0077 JMPBCK:          ..JUMP BACK TO LNSET,
7611 ;          0078                                  ..(THEN TO FFT MAIN)
7611 D5;          0079          EXIT
7612 ;          0080

```

ASP FFT Program  
Run Time Code

```

7612 ;          0081 ..*****
7612 ;          0082 ..FRAM PRINT TABLES
7612 ;          0083
7612 ;          0084          ORG #23BE          ..WORKER GAIN
23BE 306231003062; 0085          ,#3062; ,#3100; ,#3062
23C4 0002;        0086          ,#0002
23C6 ;          0087
23C6 ;          0088          ORG #7400          ..FFT OUTPUT VALUES
7400 30E0310030E0; 0089          ,#30E0; ,#3100; ,#30E0
7406 00200008;    0090          ,#0020; ,#0008
740A ;          0091
740A ;          0092          ORG #7410          ..SLAST/DT
7410 304631003000; 0093          ,#3046; ,#3100; ,#3000
7416 0004;        0094          ,#0004
7418 ;          0095
7418 ;          0096          ORG #7420          ..YP/YSTBL
7420 30A531003000; 0097          ,#30A5; ,#3100; ,#3000
7426 000F;        0098          ,#000F
7428 ;          0099
7428 ;          0100          ORG #7430          ..ALL OF FFT-RAM
7430 30C431003000; 0101          ,#30C4; ,#3100; ,#3000
7436 003C;        0102          ,#003C
7438 ;          0103
7438 ;          0104          ORG #7450          ..TOTAL SHIFT LENGTH
7450 30DA310030DA; 0105          ,#30DA; ,#3100; ,#30DA
7456 0002;        0106          ,#0002
7458 ;          0107
7458 ;          0108          ORG #74E0          ..R(MA) & R(MQ)
74E0 73FA740073FA; 0109          ,#73FA; ,#7400; ,#73FA
74E6 0004;        0110          ,#0004
74E8 ;          0111
74E8 ;          0112          ORG #74E8          ..Y,X,WORKING LOWER 2/3
74E8 30FB310030FB; 0113          ,#30FB; ,#3100; ,#30FB
74EE 0002;        0114          ,#0002
74F0 ;          0115
74F0 ;          0116          ORG #7520          ..R(MQ.0), R(AC)
7520 73FD740073FD; 0117          ,#73FD; ,#7400; ,#73FD
7526 0003;        0118          ,#0003
7528 ;          0119
7528 ;          0120          ORG #7528          ..GAMMA INT/FRC
7528 30EC310030EC; 0121          ,#30EC; ,#3100; ,#30EC
752E 0004;        0122          ,#0004
7530 ;          0123
7530 ;          0124          ORG #7530          ..R(ZAP3)
7530 73F6740073F6; 0125          ,#73F6; ,#7400; ,#73F6
7536 0002;        0126          ,#0002
7538 ;          0127
7538 ;          0128          ORG #7538          ..MAX.ENERGY.INT/EXP
7538 30F0310030F0; 0129          ,#30F0; ,#3100; ,#30F0
753E 0004;        0130          ,#0004
7540 ;          0131

```

```

7540 ; 0132 ..#####
7540 ; 0133 ..NOW DO FFT PROGRAM BREAKS
7540 ; 0134
7540 ; 0135 ..AFTER LNSET, PRINT FRAM CHUNKS & INPUT DATA
7540 ; 0136 ORG #1878
1878 D4713D74107416;0137 CALL PRMEM; ,#7410; ,#7416
187F D4713D74207426;0138 CALL PRMEM; ,#7420; ,#7426
1886 D4713D74307436;0139 CALL PRMEM; ,#7430; ,#7436
188D D47600; 0140 CALL INPRNT
1890 D5; 0141 EXIT
1891 ; 0142
1891 ; 0143 ..AFTER SCALE, PRINT INPUT FIFO
1891 ; 0144 ORG #18EA
18EA D47600; 0145 CALL INPRNT
18ED D4713D74507456;0146 CALL PRMEM; ,#7450; ,#7456
18F4 D5; 0147 EXIT
18F5 ; 0148
18F5 ; 0149 ..AFTER P-SHAPE, EXIT BACK
18F5 ; 0150 ORG #196C
196C D5; 0151 EXIT
196D ; 0152
196D ; 0153 ..AFTER S-SHAPE, EXIT BACK
196D ; 0154 ORG #1A2B
1A2B D5; 0155 EXIT
1A2C ; 0156
1A2C ; 0157 ..DURING FFT, AFTER NEWCOL, PRINT FFT FIFO
1A2C ; 0158 ORG #1AA7
1AA7 C07800; 0159 LBR #7800
1AAA ; 0160
1AAA ; 0161 ORG #7800
7800 D47015; 0162 CALL FFTDTA
7803 F8E9A8; 0163 ,#F8E9A8
7806 C01AAA; 0164 LBR #1AAA
7809 ; 0165
7809 ; 0166 ..DURING FFT, AFTER MAG CHECK, PRINT FFT FIFO
7809 ; 0167 ORG #1AD7
1AD7 C07900; 0168 LBR #7900
1ADA ; 0169
1ADA ; 0170 ORG #7900
7900 F829B0; 0171 ,#F829B0
7903 F860A0D0; 0172 ,#F860A0D0
7907 D47015; 0173 CALL FFTDTA
790A C01ADE; 0174 LBR #1ADE
790D ; 0175
790D ; 0176 ..AFTER FFT, PRINT FFT DATA
790D ; 0177 ORG #1BD3
1BD3 D47015; 0178 CALL FFTDTA
1BD6 D5; 0179 EXIT
1BD7 ; 0180

```

ASP FFT Program  
Run Time Code

1BD7 ;	0181	..AFTER UNSCRM, PRINT FFT FIFO
1BD7 ;	0182	ORG #1CCD
1CCD D47015;	0183	CALL FFTDTA
1CD0 D5;	0184	EXIT
1CD1 ;	0185	
1CD1 ;	0186	..DURING ORDER, AFTER SCALE, PRINT FFT FIFO
1CD1 ;	0187	ORG #1D62
1D62 D47A00;	0188	CALL #7A00
1D65 ;	0189	
1D65 ;	0190	ORG #7A00
7A00 D47015;	0191	CALL FFTDTA
7A03 F8D8A8;	0192	,#F8D8A8
7A06 D5;	0193	EXIT
7A07 ;	0194	
7A07 ;	0195	..AFTER ORDER, PRINT FFT FIFO
7A07 ;	0196	ORG #1F7A
1F7A D47015;	0197	CALL FFTDTA
1F7D D5;	0198	EXIT
1F7E ;	0199	
1F7E ;	0200	..DURING MAGAPX, AFTER SCALE, PRINT FFT FIFO
1F7E ;	0201	ORG #2019
2019 D47015;	0202	CALL FFTDTA
201C ;	0203	
201C ;	0204	..AFTER MAGAPX, PRINT FFT OUTPUT DATA
201C ;	0205	ORG #20D4
20D4 D47015;	0206	CALL FFTDTA
20D7 D5;	0207	EXIT
20D8 ;	0208	
20D8 ;	0209	..DURING SMOOTH/MAX, AFTER SCALE, PRINT FFT FIFO
20D8 ;	0210	ORG #2119
2119 D47015;	0211	CALL FFTDTA
211C ;	0212	
211C ;	0213	..AFTER SMOOTH/MAX
211C ;	0214	..PRINT FFT DATA & SOME OUTPUT VALUES
211C ;	0215	ORG #2201
2201 C02230;	0216	LBR #2230
2204 ;	0217	
2204 ;	0218	ORG #2230
2230 D47015;	0219	CALL FFTDTA
2233 D4713D74007408;	0220	CALL PRMEM; ,#7400; ,#7408
223A D5;	0221	EXIT
223B ;	0222	
223B ;	0223	..DURING CKFMAX, DURING SPECTRA CORRECT,
223B ;	0224	..PRINT ADDRESSES OF OPERANDS
223B ;	0225	ORG #229B
229B C07A20;	0226	LBR #7A20
229E ;	0227	
229E ;	0228	ORG #7A20
7A20 D4713D74E074E6;	0229	CALL PRMEM; ,#74E0; ,#74E6
7A27 F804A0D0;	0230	,#F804A0; ,#D0
7A2B C0229F;	0231	LBR #229F



```

7A2E ;                0232
7A2E ;                0233 ..THEN PRINT RESULT
7A2E ;                0234     ORG #22A8
22A8 C07A30;          0235     LBR #7A30
22AB ;                0236
22AB ;                0237     ORG #7A30
7A30 D4713D75307536; 0238     CALL PRMEM; ,#7530; ,#7536
7A37 F823BE;          0239     ,#F823BE
7A3A C022AB;          0240     LBR #22AB
7A3D ;                0241
7A3D ;                0242 ..PRINT RESULT OF INSFIX NORMALIZE MULTIPLY
7A3D ;                0243     ORG #22BD
22BD C07A40;          0244     LBR #7A40
22C0 ;                0245
22C0 ;                0246     ORG #7A40
7A40 D4713D7538753E; 0247     CALL PRMEM; ,#7538; ,#753E
7A47 4DBB4D;          0248     ,#4DBB4D
7A4A C022C0;          0249     LBR #22C0
7A4D ;                0250
7A4D ;                0251 ..PRINT RESULT OF LENGTH FIX
7A4D ;                0252     ORG #22D6
22D6 C07A50;          0253     LBR #7A50
22D9 ;                0254
22D9 ;                0255     ORG #7A50
7A50 D4713D7538753E; 0256     CALL PRMEM; ,#7538; ,#753E
7A57 F830BDBF;        0257     ,#F830BDBF
7A5B C022DA;          0258     LBR #22DA
7A5E ;                0259
7A5E ;                0260 ..PRINT RESULT OF GEOPHONE GAIN
7A5E ;                0261     ORG #22F6
22F6 C07A60;          0262     LBR #7A60
22F9 ;                0263
22F9 ;                0264     ORG #7A60
7A60 D4713D7538753E; 0265     CALL PRMEM; ,#7538; ,#753E
7A67 F862A8;          0266     ,#F862A8
7A6A C022F9;          0267     LBR #22F9
7A6D ;                0268
7A6D ;                0269 ..PRINT RESULT OF AMP GAIN
7A6D ;                0270     ORG #2321
2321 C07A70;          0271     LBR #7A70
2324 ;                0272
2324 ;                0273     ORG #7A70
7A70 D4713D23BE23C4; 0274     CALL PRMEM; ,#23BE; ,#23C4 ..PRINT GAIN
7A77 D4713D7538753E; 0275     CALL PRMEM; ,#7538; ,#753E
7A7E F8E2A8;          0276     ,#F8E2A8
7A81 C02324;          0277     LBR #2324
7A84 ;                0278
7A84 ;                0279 ..AFTER CKFMAX, PRINT OUTPUT VALUES
7A84 ;                0280     ORG #234E
234E D47500;          0281     CALL #7500
2351 D5;              0282     EXIT

```

ASP FFT Program  
Run Time Code

2352 ;	0283	
2352 ;	0284	..DURING GMACPT
2352 ;	0285	..PRINT Y & INSFIX ADDRESSES
2352 ;	0286	ORG #2456
2456 C07580;	0287	LBR #7580
2459 ;	0288	
2459 ;	0289	ORG #7580
7580 D4713D74E074E6;	0290	CALL PRMEM; ,#74E0; ,#74E6
7587 D01D1D;	0291	,#D0; ,#1D1D
758A C02459;	0292	LBR #2459
758D ;	0293	
758D ;	0294	..PRINT PRODUCT RESULTING
758D ;	0295	ORG #2465
2465 C07A10;	0296	LBR #7A10
2468 ;	0297	
2468 ;	0298	ORG #7A10
7A10 D4713D75307536;	0299	CALL PRMEM; ,#7530; ,#7536
7A17 E88BF473;	0300	,#E8; ,#8BF473
7A1B C02469;	0301	LBR #2469
7A1E ;	0302	
7A1E ;	0303	..PRINT AVG(Y) AND LOG(AVG(Y))
7A1E ;	0304	ORG #2492
2492 C07598;	0305	LBR #7598
2495 ;	0306	
2495 ;	0307	ORG #7598
7598 D4713D74E874EE;	0308	CALL PRMEM; ,#74E8; ,#74EE
759F F825B0;	0309	,#F825B0
75A2 F8D0A0;	0310	,#F8D0A0
75A5 D0;	0311	,#D0
75A6 D4713D75207526;	0312	CALL PRMEM; ,#7520; ,#7526
75AD C02499;	0313	LBR #2499
75B0 ;	0314	
75B0 ;	0315	..PRINT X2 & LOG(X2)
75B0 ;	0316	ORG #24E6
24E6 C076A0;	0317	LBR #76A0
24E9 ;	0318	
24E9 ;	0319	ORG #76A0
76A0 D4713D74E874EE;	0320	CALL PRMEM; ,#74E8; ,#74EE
76A7 D0;	0321	,#D0
76A8 D4713D75207526;	0322	CALL PRMEM; ,#7520; ,#7526
76AF F8F9A8;	0323	,#F8F9A8
76B2 C024EA;	0324	LBR #24EA
76B5 ;	0325	

```

76B5 ;          0326 ..PRINT X1 & LOG(X1)
76B5 ;          0327      ORG #24FC
24FC C075B8;    0328      LBR #75B8
24FF ;          0329
24FF ;          0330      ORG #75B8
75B8 D4713D74E874EE;0331      CALL PRMEM; ,#74E8; ,#74EE
75BF D0;        0332      ,#D0
75C0 D4713D75207526;0333      CALL PRMEM; ,#7520; ,#7526
75C7 F8F9A8;    0334      ,#F8F9A8
75CA C02500;    0335      LBR #2500
75CD ;          0336
75CD ;          0337 ..PRINT DIVISION RESULT (HEX)
75CD ;          0338      ORG #2567
2567 C075D0;    0339      LBR #75D0
256A ;          0340
256A ;          0341      ORG #75D0
75D0 D4713D7528752E;0342      CALL PRMEM; ,#7528; ,#752E
75D7 F800AA;    0343      ,#F800AA
75DA C0256A;    0344      ,#C0256A
75DD ;          0345
75DD ;          0346 ..AT END OF GMACPT PRINT RESULTS
75DD ;          0347      ORG #25CB
25CB D47500;    0348      CALL #7500
25CE D5;        0349      EXIT
25CF ;          0350
25CF ;          0351 ..AT END OF FOLPL PRINT RESULTS
25CF ;          0352      ORG #2CA5
2CA5 D47500;    0353      CALL #7500
2CA8 D5;        0354      EXIT
2CA9 ;          0355
2CA9 ;          0356 ..#####
2CA9 ;          0357 ..END OF FFT COMPLETE TRACE.SR

```

ASP FFT Program  
Run Time Code

B.1.4 FFT OUT TRACE

```

0000 ;          0001
0000 ;          0002 ..FFT OUT TRACE.SR          9 MARCH 80          9:45 AM
0000 ;          0003
0000 ;          0004          ZAP=#00; PC=#03; FRP=#08; MP=#09
0000 ;          0005          ZAP2=#0A; ZAP3=#0B; ZAP1=#0C
0000 ;          0006          MA=#0D; MQ=#0E; AC=#0F
0000 ;          0007
0000 ;          0008 ..*****
0000 ;          0009 ..SET P-LENGTH = 64, S-LENGTH = 128
0000 ;          0010          ORG #3046
3046 11110000; 0011          ,#11110000          ..SET SLAST NOT #0000
304A ;          0012          ..          DT=#0000
304A ;          0013
304A ;          0014 ..SET GAIN = 1          ..(MAY BE CHANGED LATER)
304A ;          0015          ORG #3062
3062 0001;     0016          ,#0001
3064 ;          0017
3064 ;          0018 ..LDFIFO SET
3064 ;          0019          ORG #51D2
51D2 C0701A;   0020          ,#C0701A          ..$P5152 FOR FIFO INPUT
51D5 ;          0021          ORG #521B
521B C0701A;   0022          ,#C0701A          ..$P51D5 FOR SINE
521E ;          0023          ORG #5232
5232 3601522A; 0024          ,#3601; ,#522A          ..(#3A01,522E FOR S-FFT)
5236 7FFF00040000; 0025          ,#7FFF; ,#0004; ,#0000
523C 000003000000; 0026          ,#0000; ,#0300; ,#0000
5242 ABAB;     0027          ,#ABAB
5244 ;          0028          ORG #5226
5226 521E0020; 0029          ,#521E; ,#0020          ..(#5222 FOR S-FFT)
522A ;          0030
522A ;          0031 ..AFTER SYSRUN INITIALIZE, GOTO WORKER
522A ;          0032 ..(MAY BE CHANGED LATER)
522A ;          0033          ORG #706D
706D C00600;   0034          LBR #0600          ..CALL WORKER
7070 ;          0035          ..(#D41700 TO RUN P-FFT)
7070 ;          0036          ..(#D41703 TO RUN S-FFT)
7070 ;          0037

```



```

7070 ;          0038 ..*****
7070 ;          0039 ..EXTERNAL SUBROUTINES
7070 ;          0040          RGDATA=#7009          ..PRINT REGISTER DATA
7070 ;          0041          PFTDTA=#700F          ..PRINT P-FFT INPUT DATA
7070 ;          0042          SFTDTA=#7012          ..PRINT S-FFT INPUT DATA
7070 ;          0043          FFTDTA=#7015          ..PRINT FFT OUTPUT DATA
7070 ;          0044          ..(OR INTERMEDIATE RESULTS)
7070 ;          0045          PRMEM=#713D          ..PRINT MEMORY
7070 ;          0046
7070 ;          0047 ..SWITCH FFTDTA PRINT TO INCLUDE TOTLSH
7070 ;          0048 ..(DOES KILL "FFTRAM" PRINT, BUT THAT'S OK)
7070 ;          0049          ORG #712E
712E D4713D74507456;0050          CALL PRMEM; ,#7450; ,#7456
7135 D5;          0051          EXIT
7136 ;          0052
7136 ;          0053 ..MAKE A FFT OUTPUT PRINT (TO BE "CALL"ED)
7136 ;          0054          ORG #7500
7500 D4713D74007406;0055          CALL PRMEM; ,#7400; ,#7406
7507 D5;          0056          EXIT
7508 ;          0057
7508 ;          0058 ..-----
7508 ;          0059 ..ROUTINE TO BE CALLED FOR INPUT DATA FIFO
7508 ;          0060
7508 ;          0061          ORG #7600
7600 ;          0062 INPRNT:          ..INPUT PRINT
7600 ;          0063 ..CHECK FOR P-FFT
7600 F830BD;          0064          LDI #30; PHI RD
7603 F8C4AD;          0065          LDI #C4; PLO RD
7606 OD320E;          0066          LDN RD; BZ ISSFFT ..IF PFTFLG=#00, THEN
7609 ;          0067          ..GOTO PRINT S-FFT DATA
7609 ;          0068
7609 ;          0069 ..ELSE PRINT P-FFT DATA
7609 D4700F;          0070          CALL PFTDTA
760C 3011;          0071          BR JMPBCK          ..THEN JUMP BACK
760E ;          0072
760E ;          0073 ISSFFT:          ..IS S-FFT
760E ;          0074 ..PRINT S-FFT DATA
760E D47012;          0075          CALL SFTDTA
7611 ;          0076
7611 ;          0077 JMPBCK:          ..JUMP BACK TO LNSET,
7611 ;          0078          ..(THEN TO FFT MAIN)
7611 D5;          0079          EXIT
7612 ;          0080

```

ASP FFT Program  
Run Time Code

```

7612 ; 0081 ..*****
7612 ; 0082 ..FRAM PRINT TABLES
7612 ; 0083
7612 ; 0084 ORG #23BE ..WORKER GAIN
23BE 306231003062; 0085 ,#3062; ,#3100; ,#3062
23C4 0002; 0086 ,#0002
23C6 ; 0087
23C6 ; 0088 ORG #7400 ..FFT OUTPUT VALUES
7400 30E0310030E0; 0089 ,#30E0; ,#3100; ,#30E0
7406 00200008; 0090 ,#0020; ,#0008
740A ; 0091
740A ; 0092 ORG #7410 ..SLAST/DT
7410 304631003000; 0093 ,#3046; ,#3100; ,#3000
7416 0004; 0094 ,#0004
7418 ; 0095
7418 ; 0096 ORG #7420 ..YP/YSTBL
7420 30A531003000; 0097 ,#30A5; ,#3100; ,#3000
7426 000F; 0098 ,#000F
7428 ; 0099
7428 ; 0100 ORG #7430 ..ALL OF FFT-RAM
7430 30C431003000; 0101 ,#30C4; ,#3100; ,#3000
7436 003C; 0102 ,#003C
7438 ; 0103
7438 ; 0104 ORG #7450 ..TOTAL SHIFT LENGTH
7450 30DA310030DA; 0105 ,#30DA; ,#3100; ,#30DA
7456 0002; 0106 ,#0002
7458 ; 0107
7458 ; 0108 ORG #74E0 ..R(MA) & R(MQ)
74E0 73FA740073FA; 0109 ,#73FA; ,#7400; ,#73FA
74E6 0004; 0110 ,#0004
74E8 ; 0111
74E8 ; 0112 ORG #74E8 ..Y,X,WORKING LOWER 2/3
74E8 30FB310030FB; 0113 ,#30FB; ,#3100; ,#30FB
74EE 0002; 0114 ,#0002
74F0 ; 0115
74F0 ; 0116 ORG #7520 ..R(MQ.O), R(AC)
7520 73FD740073FD; 0117 ,#73FD; ,#7400; ,#73FD
7526 0003; 0118 ,#0003
7528 ; 0119
7528 ; 0120 ORG #7528 ..GAMMA INT/FRC
7528 30EC310030EC; 0121 ,#30EC; ,#3100; ,#30EC
752E 0004; 0122 ,#0004
7530 ; 0123
7530 ; 0124 ORG #7530 ..R(ZAP3)
7530 73F6740073F6; 0125 ,#73F6; ,#7400; ,#73F6
7536 0002; 0126 ,#0002
7538 ; 0127
7538 ; 0128 ORG #7538 ..MAX.ENERGY.INT/EXP
7538 30F0310030F0; 0129 ,#30F0; ,#3100; ,#30F0
753E 0004; 0130 ,#0004
7540 ; 0131

```

```

7540 ;          0132 ..#####
7540 ;          0133 ..NOW DO FFT PROGRAM BREAKS
7540 ;          0134
7540 ;          0135 ..AFTER LNSET, EXIT
7540 ;          0136         ORG #1878
1878 D5;        0137         EXIT
1879 ;          0138
1879 ;          0139 ..AFTER SCALE, EXIT
1879 ;          0140         ORG #18EA
18EA D5;        0141         EXIT
18EB ;          0142
18EB ;          0143 ..AFTER P-SHAPE, EXIT BACK
18EB ;          0144         ORG #196C
196C D5;        0145         EXIT
196D ;          0146
196D ;          0147 ..AFTER S-SHAPE, EXIT BACK
196D ;          0148         ORG #1A2B
1A2B D5;        0149         EXIT
1A2C ;          0150
1A2C ;          0151 ..DURING FFT, AFTER NEWCOL, CONTINUE
1A2C ;          0152         ORG #1AA7
1AA7 F8E9A8;    0153         ,#F8E9A8
1AAA ;          0154
1AAA ;          0155 ..DURING FFT, AFTER MAG CHECK, CONTINUE
1AAA ;          0156         ORG #1AD7
1AD7 F829B0;    0157         ,#F829B0
1ADA ;          0158
1ADA ;          0159 ..AFTER FFT, EXIT
1ADA ;          0160         ORG #1BD3
1BD3 D5;        0161         EXIT
1BD4 ;          0162
1BD4 ;          0163 ..AFTER UNSCRM, EXIT
1BD4 ;          0164         ORG #1CCD
1CCD D5;        0165         EXIT
1CCE ;          0166
1CCE ;          0167 ..DURING ORDER, AFTER SCALE, CONTINUE
1CCE ;          0168         ORG #1D62
1D62 F8D8A8;    0169         ,#F8D8A8
1D65 ;          0170
1D65 ;          0171 ..AFTER ORDER, EXIT
1D65 ;          0172         ORG #1F7A
1F7A D5;        0173         EXIT
1F7B ;          0174
1F7B ;          0175 ..DURING MAGAPX, AFTER SCALE, CONTINUE
1F7B ;          0176         ORG #2019
2019 C4C4C4;    0177         ,#C4C4C4
201C ;          0178
201C ;          0179 ..AFTER MAGAPX, PRINT FFT OUTPUT DATA
201C ;          0180         ORG #20D4
20D4 D47015;    0181         CALL FFTDTA
20D7 D5;        0182         EXIT

```

ASP FFT Program  
Run Time Code

```

20D8 ;                0183
20D8 ;                0184 ..DURING SMOOTH/MAX, AFTER SCALE, CONTINUE
20D8 ;                0185             ORG #2119
2119 C4C4C4;          0186             ,#C4C4C4
211C ;                0187
211C ;                0188 ..AFTER SMOOTH/MAX
211C ;                0189 ..PRINT FFT DATA & SOME OUTPUT VALUES
211C ;                0190             ORG #2201
2201 C02230;          0191             LBR #2230
2204 ;                0192
2204 ;                0193             ORG #2230
2230 D47015;          0194             CALL FFTDTA
2233 D4713D74007408; 0195             CALL PRMEM; ,#7400; ,#7408
223A D5;              0196             EXIT
223B ;                0197
223B ;                0198 ..DURING CKFMAX, DURING SPECTRA CORRECT,
223B ;                0199 ..PRINT ADDRESSES OF OPERANDS
223B ;                0200             ORG #229B
229B C07A20;          0201             LBR #7A20
229E ;                0202
229E ;                0203             ORG #7A20
7A20 D4713D74E074E6; 0204             CALL PRMEM; ,#74E0; ,#74E6
7A27 F804A0D0;        0205             ,#F804A0; ,#D0
7A2B C0229F;          0206             LBR #229F
7A2E ;                0207
7A2E ;                0208 ..THEN PRINT RESULT
7A2E ;                0209             ORG #22A8
22A8 C07A30;          0210             LBR #7A30
22AB ;                0211
22AB ;                0212             ORG #7A30
7A30 D4713D75307536; 0213             CALL PRMEM; ,#7530; ,#7536
7A37 F823BE;          0214             ,#F823BE
7A3A C022AB;          0215             LBR #22AB
7A3D ;                0216
7A3D ;                0217 ..PRINT RESULT OF INSFIX NORMALIZE MULTIPLY
7A3D ;                0218             ORG #22BD
22BD C07A40;          0219             LBR #7A40
22C0 ;                0220
22C0 ;                0221             ORG #7A40
7A40 D4713D7538753E; 0222             CALL PRMEM; ,#7538; ,#753E
7A47 4DBB4D;          0223             ,#4DBB4D
7A4A C022C0;          0224             LBR #22C0
7A4D ;                0225
7A4D ;                0226 ..PRINT RESULT OF LENGTH FIX
7A4D ;                0227             ORG #22D6
22D6 C07A50;          0228             LBR #7A50
22D9 ;                0229
22D9 ;                0230             ORG #7A50
7A50 D4713D7538753E; 0231             CALL PRMEM; ,#7538; ,#753E
7A57 F830BDBF;        0232             ,#F830BDBF
7A5B C022DA;          0233             LBR #22DA

```



```

7A5E ;                0234
7A5E ;                0235 ..PRINT RESULT OF GEOPHONE GAIN
7A5E ;                0236     ORG #22F6
22F6 C07A60;          0237     LBR #7A60
22F9 ;                0238
22F9 ;                0239     ORG #7A60
7A60 D4713D7538753E;0240     CALL PRMEM; ,#7538; ,#753E
7A67 F862A8;          0241     ,#F862A8
7A6A C022F9;          0242     LBR #22F9
7A6D ;                0243
7A6D ;                0244 ..PRINT RESULT OF AMP GAIN
7A6D ;                0245     ORG #2321
2321 C07A70;          0246     LBR #7A70
2324 ;                0247
2324 ;                0248     ORG #7A70
7A70 D4713D23BE23C4;0249     CALL PRMEM; ,#23BE; ,#23C4 ..PRINT GAIN
7A77 D4713D7538753E;0250     CALL PRMEM; ,#7538; ,#753E
7A7E F8E2A8;          0251     ,#F8E2A8
7A81 C02324;          0252     LBR #2324
7A84 ;                0253
7A84 ;                0254 ..AFTER CKFMAX, PRINT OUTPUT VALUES
7A84 ;                0255     ORG #234E
234E D47500;          0256     CALL #7500
2351 D5;              0257     EXIT
2352 ;                0258
2352 ;                0259 ..DURING GMACPT
2352 ;                0260 ..PRINT Y & INSFIX ADDRESSES
2352 ;                0261     ORG #2456
2456 C07580;          0262     LBR #7580
2459 ;                0263
2459 ;                0264     ORG #7580
7580 D4713D74E074E6;0265     CALL PRMEM; ,#74E0; ,#74E6
7587 D01D1D;          0266     ,#D0; ,#1D1D
758A C02459;          0267     LBR #2459
758D ;                0268
758D ;                0269 ..PRINT PRODUCT RESULTING
758D ;                0270     ORG #2465
2465 C07A10;          0271     LBR #7A10
2468 ;                0272
2468 ;                0273     ORG #7A10
7A10 D4713D75307536;0274     CALL PRMEM; ,#7530; ,#7536
7A17 E88BF473;        0275     ,#E8; ,#8BF473
7A1B C02469;          0276     LBR #2469
7A1E ;                0277

```

ASP FFT Program  
Run Time Code

```

7A1E ;                0278 ..PRINT AVG(Y) AND LOG(AVG(Y))
7A1E ;                0279     ORG #2492
2492 C07598;          0280     LBR #7598
2495 ;                0281
2495 ;                0282     ORG #7598
7598 D4713D74E874EE;0283     CALL PRMEM; ,#74E8; ,#74EE
759F F825B0;          0284     ,#F825B0
75A2 F8DOA0;          0285     ,#F8DOA0
75A5 D0;              0286     ,#D0
75A6 D4713D75207526;0287     CALL PRMEM; ,#7520; ,#7526
75AD C02499;          0288     LBR #2499
75B0 ;                0289
75B0 ;                0290 ..PRINT X2 & LOG(X2)
75B0 ;                0291     ORG #24E6
24E6 C076A0;          0292     LBR #76A0
24E9 ;                0293
24E9 ;                0294     ORG #76A0
76A0 D4713D74E874EE;0295     CALL PRMEM; ,#74E8; ,#74EE
76A7 D0;              0296     ,#D0
76A8 D4713D75207526;0297     CALL PRMEM; ,#7520; ,#7526
76AF F8F9A8;          0298     ,#F8F9A8
76B2 C024EA;          0299     LBR #24EA
76B5 ;                0300
76B5 ;                0301 ..PRINT X1 & LOG(X1)
76B5 ;                0302     ORG #24FC
24FC C075B8;          0303     LBR #75B8
24FF ;                0304
24FF ;                0305     ORG #75B8
75B8 D4713D74E874EE;0306     CALL PRMEM; ,#74E8; ,#74EE
75BF D0;              0307     ,#D0
75C0 D4713D75207526;0308     CALL PRMEM; ,#7520; ,#7526
75C7 F8F9A8;          0309     ,#F8F9A8
75CA C02500;          0310     LBR #2500
75CD ;                0311
75CD ;                0312 ..PRINT DIVISION RESULT (HEX)
75CD ;                0313     ORG #2567
2567 C075D0;          0314     LBR #75D0
256A ;                0315
256A ;                0316     ORG #75D0
75D0 D4713D7528752E;0317     CALL PRMEM; ,#7528; ,#752E
75D7 F800AA;          0318     ,#F800AA
75DA C0256A;          0319     ,#C0256A
75DD ;                0320
75DD ;                0321 ..AT END OF GMACPT PRINT RESULTS
75DD ;                0322     ORG #25CB
25CB D47500;          0323     CALL #7500
25CE D5;              0324     EXIT
25CF ;                0325

```

ASP FFT Program  
Run Time Code

```

25CF ;          0326 ..AT END OF FOLPL PRINT RESULTS
25CF ;          0327          ORG #2CA5
2CA5 D47500;    0328          CALL #7500
2CA8 D5;        0329          EXIT
2CA9 ;          0330
2CA9 ;          0331 ..#####
2CA9 ;          0332 ..END OF FFT OUT TRACE.SR

```

ASP FFT Program  
Run Time Code

B.1.5 FFT NO TRACE

```

0000 ;          0001
0000 ;          0002 ..FFT NO TRACE.SR          5 MARCH 80          3:30 PM
0000 ;          0003
0000 ;          0004          ZAP=#00; PC=#03; FRP=#08; MP=#09
0000 ;          0005          ZAP2=#0A; ZAP3=#0B; ZAP1=#0C
0000 ;          0006          MA=#0D; MQ=#0E; AC=#0F
0000 ;          0007
0000 ;          0008 ..*****
0000 ;          0009 ..NOW DO FFT PROGRAM CLEARS
0000 ;          0010
0000 ;          0011 ..AFTER LNSET, EXIT
0000 ;          0012          ORG #1878
1878 D5;        0013          EXIT
1879 ;          0014
1879 ;          0015 ..AFTER SCALE, EXIT
1879 ;          0016          ORG #18EA
18EA D5;        0017          EXIT
18EB ;          0018
18EB ;          0019 ..AFTER P-SHAPE, EXIT BACK
18EB ;          0020          ORG #196C
196C D5;        0021          EXIT
196D ;          0022
196D ;          0023 ..AFTER S-SHAPE, EXIT BACK
196D ;          0024          ORG #1A2B
1A2B D5;        0025          EXIT
1A2C ;          0026
1A2C ;          0027 ..DURING FFT, AFTER NEWCOL, CONTINUE
1A2C ;          0028          ORG #1AA7
1AA7 F8E9A8;    0029          ,#F8E9A8
1AAA ;          0030
1AAA ;          0031 ..DURING FFT, AFTER MAG CHECK, CONTINUE
1AAA ;          0032          ORG #1AD7
1AD7 F829B0;    0033          ,#F829B0
1ADA ;          0034
1ADA ;          0035 ..AFTER FFT, EXIT
1ADA ;          0036          ORG #1BD3
1BD3 D5;        0037          EXIT
1BD4 ;          0038
1BD4 ;          0039 ..AFTER UNSCRM, EXIT
1BD4 ;          0040          ORG #1CCD
1CCD D5;        0041          EXIT
1CCE ;          0042
1CCE ;          0043 ..DURING ORDER, AFTER SCALE, CONTINUE
1CCE ;          0044          ORG #1D62
1D62 F8D8A8;    0045          ,#F8D8A8

```



1D65 ;	0046
1D65 ;	0047 ..AFTER ORDER, EXIT
1D65 ;	0048       ORG #1F7A
1F7A D5;	0049       EXIT
1F7B ;	0050
1F7B ;	0051 ..DURING MAGAPX, AFTER SCALE, CONTINUE
1F7B ;	0052       ORG #2019
2019 C4C4C4;	0053       , #C4C4C4
201C ;	0054
201C ;	0055 ..AFTER MAGAPX, EXIT
201C ;	0056       ORG #20D4
20D4 D5;	0057       EXIT
20D5 ;	0058
20D5 ;	0059 ..DURING SMOOTH/MAX, AFTER SCALE, CONTINUE
20D5 ;	0060       ORG #2119
2119 C4C4C4;	0061       , #C4C4C4
211C ;	0062
211C ;	0063 ..AFTER SMOOTH/MAX, EXIT
211C ;	0064       ORG #2201
2201 D5;	0065       EXIT
2202 ;	0066
2202 ;	0067 ..DURING CKFMAX, DURING SPECTRA CORRECT, CONTINUE
2202 ;	0068       ORG #229B
229B F804A0D0;	0069       , #F804A0; , #D0
229F ;	0070
229F ;	0071 ..THEN CONTINUE
229F ;	0072       ORG #22A8
22A8 F823BE;	0073       , #F823BE
22AB ;	0074
22AB ;	0075 ..CONTINUE
22AB ;	0076       ORG #22BD
22BD 4DBB4D;	0077       , #4DBB4D
22C0 ;	0078
22C0 ;	0079 ..CONTINUE
22C0 ;	0080       ORG #22D6
22D6 F830BDBF;	0081       , #F830BDBF
22DA ;	0082
22DA ;	0083 ..CONTINUE
22DA ;	0084       ORG #22F6
22F6 F862A8;	0085       , #F862A8
22F9 ;	0086
22F9 ;	0087 ..CONTINUE
22F9 ;	0088       ORG #2321
2321 F8E2A8;	0089       , #F8E2A8
2324 ;	0090
2324 ;	0091 ..AFTER CKFMAX, EXIT
2324 ;	0092       ORG #234E
234E D5;	0093       EXIT
234F ;	0094

ASP FFT Program  
Run Time Code

234F ;	0095 ..DURING GMACPT, CONTINUE
234F ;	0096 ORG #2456
2456 D01D1D;	0097 ,#D0; ,#1D1D
2459 ;	0098
2459 ;	0099 ..CONTINUE
2459 ;	0100 ORG #2465
2465 E88BF473;	0101 ,#E8; ,#8BF473
2469 ;	0102
2469 ;	0103 ..CONTINUE
2469 ;	0104 ORG #2492
2492 F825B0;	0105 ,#F825B0
2495 ;	0106
2495 ;	0107 ..CONTINUE
2495 ;	0108 ORG #24E6
24E6 D0;	0109 ,#D0
24E7 F8F9A8;	0110 ,#F8F9A8
24EA ;	0111
24EA ;	0112 ..CONTINUE
24EA ;	0113 ORG #24FC
24FC D0;	0114 ,#D0
24FD F8F9A8;	0115 ,#F8F9A8
2500 ;	0116
2500 ;	0117 ..CONTINUE
2500 ;	0118 ORG #2567
2567 F800AA;	0119 ,#F800AA
256A ;	0120
256A ;	0121 ..AT END OF GMACPT EXIT
256A ;	0122 ORG #25CB
25CB D5;	0123 EXIT
25CC ;	0124
25CC ;	0125 ..AT END OF F0LPL PRINT RESULTS
25CC ;	0126 ORG #2CA5
2CA5 D5;	0127 EXIT
2CA6 ;	0128
2CA6 ;	0129 ..#####
2CA6 ;	0130 ..END OF FFT NO TRACE.SR

B.1.6 MULTI FFT

```

0000 ;          0001
0000 ;          0002 ..MULTI FFT.SR          9 MAY 80          3:40 PM
0000 ;          0003
0000 ;          0004 ..FROM SYSRUN, GOTO HANDLER
0000 ;          0005          ORG #706D
706D C06000;    0006          LBR #6000
7070 ;          0007
7070 ;          0008 ..HANDLER
7070 ;          0009          ORG #6000
6000 ;          0010
6000 ;          0011 ..INITIALIZE
6000 ;          0012 ..SET START POINTER @ 6100 TO #3601 (#3A01)
6000 F861BF;    0013          LDI #61; PHI RF
6003 F800AF;    0014          LDI #00; PLO RF
6006 F8365F1F;  0015          LDI #36; STR RF; INC RF  ..(#3A)
600A F8005F;    0016          LDI #00; STR RF
600D ;          0017
600D ;          0018 FFTLP:          ..DO FFT LOOP
600D ;          0019 ..SET DT = #1111, SLAST = #0000
600D F830BF;    0020          LDI #30; PHI RF
6010 F846AF;    0021          LDI #46; PLO RF
6013 F8115F1F;  0022          LDI #11; STR RF; INC RF
6017 5F1F;      0023          STR RF; INC RF
6019 F8005F1F;  0024          LDI #00; STR RF; INC RF
601D 5F1F;      0025          STR RF; INC RF
601F ;          0026
601F ;          0027 ..GET STARTING OUTPUT POINTER INTO RD
601F ;          0028 ..AND PASS TO YPTBL (YSTBL)
601F F8A5AF;    0029          LDI #A5; PLO RF ..(#AE)
6022 F861BE;    0030          LDI #61; PHI RE
6025 F800AE;    0031          LDI #00; PLO RE
6028 4EBD5F1F;  0032          LDA RE; PHI RD; STR RF; INC RF
602C 0EAD5F;    0033          LDN RE; PLO RD; STR RF
602F ;          0034
602F ;          0035 ..REFILL FIFO WITH INPUT DATA
602F ;          0036 ..IF PFFT @ #7A00 TO #7A7F
602F ;          0037 ..IF SFFT @ #7B00 TO #7BFF
602F ;          0038 ..SET UP INPUT POINTER
602F F87ABF;    0039          LDI #7A; PHI RF ..(#7B)
6032 F800AF;    0040          LDI #00; PLO RF
6035 ;          0041
6035 ;          0042 PASVLU:          ..PASS A VALUE
6035 4F5D1D;    0043          LDA RF; STR RD; INC RD
6038 4F5D1D;    0044          LDA RF; STR RD; INC RD
603B ;          0045

```

ASP FFT Program  
Run Time Code

```

603B ; 0046 ..CHECK FOR DONE PASSING
603B 8FFF80; 0047 GLO RF; SMI #80 ..(#00)
603E 9D7F7A; 0048 GHI RD; SMBI #7A ..(#7C)
6041 C36053; 0049 LBDF CALFFT ..IF DONE, GOTO FFT
6044 ; 0050
6044 ; 0051 ..ELSE, CHECK FOR PAST FIFO END
6044 8DFF01; 0052 GLO RD; SMI #01
6047 9D7F3A; 0053 GHI RD; SMBI #3A ..(#3E)
604A CB6035; 0054 LBNF PASVLU ..IF UNDER, REPASS
604D ; 0055
604D ; 0056 ..ELSE, RESET AND THEN PASS
604D F836BD; 0057 LDI #36; PHI RD ..(#3A)
6050 C06035; 0058 LBR PASVLU
6053 ; 0059
6053 ; 0060 CALFFT: ..CALL FFT
6053 ; 0061 ..PRINT FIRST INPUT ADDRESS
6053 D4713D61106116; 0062 ,#D4713D; ,#6110; ,#6116
605A ; 0063
605A D41700; 0064 CALL #1700 ..(#1703)
605D ; 0065
605D ; 0066 ..ON RETURN, PRINT VALUES
605D D4713D61206126; 0067 ,#D4713D; ,#6120; ,#6126
6064 ; 0068
6064 ; 0069 ..THEN INCREMENT STARTING VALUE BY 2
6064 F861BF; 0070 LDI #61; PHI RF
6067 F801AFEF; 0071 LDI #01; PLO RF; SEX RF
606B F802F4AE73; 0072 LDI #02; ADD; PLO RE; STXD
6070 F80074BE5F; 0073 LDI #00; ADC; PHI RE; STR RF
6075 ; 0074
6075 ; 0075 ..CHECK FOR DONE DOING FFT'S
6075 8EFF01; 0076 GLO RE; SMI #01
6078 9E7F3A; 0077 GHI RE; SMBI #3A ..(#3E)
607B C37006; 0078 LBDF #7006 ..IF DONE, RETURN TO MONITOR
607E ; 0079
607E ; 0080 ..ELSE DO NEXT FFT
607E C0600D; 0081 LBR FFTLP
6081 ; 0082
6081 ; 0083
6081 ; 0084 ..*****
6081 ; 0085
6081 ; 0086 ..BREAK OUT OF FFTMN BEFORE PRINTING TO TERMINAL
6081 ; 0087 ORG #173D
173D D5; 0088 EXIT
173E ; 0089
173E ; 0090 ..PRINT YPTBL (YSTBL)
173E ; 0091 ORG #6110
6110 30A5310030A5; 0092 ,#30A5; ,#3100; ,#30A5 ..(#30AE)
6116 0002; 0093 ,#0002
6118 ; 0094

```



```

6118 ;          0095 ..PRINT OUTPUT VALUES
6118 ;          0096      ORG #6120
6120 30E0310030E0; 0097      ,#30E0; ,#3100; ,#30E0
6126 0020;        0098      ,#0020
6128 ;          0099
6128 ;          0100 ..#####
6128 ;          0101 ..END OF MULTI FFT.SR

```

ASP FFT Program  
Run Time Code

B.1.7 8085 FULL TRACE

0000 ;	0001		
0000 ;	0002	..8085 FULL TRACE.SR	13 MAY 80 5:45 PM
0000 ;	0003		
0000 ;	0004	ORG #1F23	
1F23 2D;	0005	,#2D	
1F24 ;	0006	ORG #1F4B	
1F4B 55;	0007	,#55	
1F4C ;	0008		
1F4C ;	0009	ORG #176D	..S-FFT BREAK
176D C0606D;	0010	LBR #606D	
1770 ;	0011		
1770 ;	0012	ORG #606D	..AFTER SFFT SEND
606D F8EBA8;	0013	LDI #EB; PLO R8	..CHECK FOR FO = 6
6070 08FD06;	0014	LDN R8; SDI #06	
6073 3292;	0015	BZ FTEXTIT	..EXIT IF EQUAL
6075 3088;	0016	BR FTPRNT	..PRINT IF NOT
6077 ;	0017		
6077 ;	0018	ORG #1796	..P-FFT BREAK
1796 C06080;	0019	LBR #6080	
1799 ;	0020		
1799 ;	0021	ORG #6080	..AFTER PFFT SEND
6080 F8EBA8;	0022	LDI #EB; PLO R8	..CHECK FOR FO = 5
6083 08FD05;	0023	LDN R8; SDI #05	
6086 3292;	0024	BZ FTEXTIT	
6088 ;	0025		
6088 D47015;	0026	FTPRT: CALL #7015	
608B D4713D74307436;	0027	CALL #713D; ,#7430; ,#7436	
6092 ;	0028		
6092 D4713D74007406;	0029	FTEXTIT: CALL #713D; ,#7400; ,#7406	
6099 D5;	0030	EXIT	
609A ;	0031		
609A ;	0032	..*****	
609A ;	0033		
609A ;	0034	ORG #712E	
712E D4713D74507456;	0035	CALL #713D; ,#7450; ,#7456	
7135 D5;	0036	EXIT	
7136 ;	0037		
7136 ;	0038	ORG #7500	
7500 D4713D74007406;	0039	CALL #713D; ,#7400; ,#7406	
7507 D5;	0040	EXIT	
7508 ;	0041		

```

7508 ;          0042 .....
7508 ;          0043      ORG #7430
7430 30C4310030C4; 0044      ,#30C4; ,#3100; ,#30C4
7436 001C;        0045      ,#001C
7438 ;          0046
7438 ;          0047      ORG #7450
7450 30DA310030DA; 0048      ,#30DA; ,#3100; ,#30DA
7456 0002;        0049      ,#0002
7458 ;          0050
7458 ;          0051      ORG #7400
7400 30E0310030E0; 0052      ,#30E0; ,#3100; ,#30E0
7406 0020;        0053      ,#0020
7408 ;          0054
7408 ;          0055 .....
7408 ;          0056
7408 ;          0057      ORG #1878      ..INPUT PRINT
1878 D47600;      0058      CALL #7600
187B D5;          0059      EXIT
187C ;          0060
187C ;          0061      ORG #7600
7600 F830BD;      0062      LDI #30; PHI RD
7603 F8C4AD;      0063      LDI #C4; PLO RD
7606 0D320E;      0064      LDN RD; BZ ISSFFT
7609 ;          0065
7609 D4700F;      0066      CALL #700F
760C 3011;        0067      ,#3011
760E ;          0068
760E D47012;      0069      ISSFFT: CALL #7012
7611 ;          0070
7611 D5;          0071      EXIT
7612 ;          0072
7612 ;          0073 .....
7612 ;          0074 .....END OF 8085 FULL TRACE.SR
7612 ;          0075

```

ASP FFT Program  
Run Time Code

B.2 RUN TIME TABLES

B.2.1 SINWAV (Sine Wave)

0000 ;	0001		
0000 ;	0002	..SINWAV.SR	9 AUG 79 4:20 PM
0000 ;	0003		
0000 ;	0004		
0000 ;	0005	..*****	
0000 ;	0006	.. SINWAV.SR	
0000 ;	0007	..	
0000 ;	0008	..SINWAV IS A LISTING OF HEX SINE VALUES	
0000 ;	0009	.. SIN (2*PI*K/256) FOR K =0 TO 255	
0000 ;	0010	..RANGING FROM ALMOST +1 (7FFF HEX)	
0000 ;	0011	.. TO ALMOST -1 (8000 HEX)	
0000 ;	0012	..IT IS FOR USE IN FFT DEBUG & TESTING	
0000 ;	0013	..*****	
0000 ;	0014		
0000 ;	0015		
0000 ;	0016		
0000 ;	0017	ORG #4000	
4000 ;	0018		
4000 ;	0019		
4000 ;	0020	SINFWA: ..SIN	
4000 0000;	0021	,#0000 ..0	
4002 0324;	0022	,#0324	
4004 0647;	0023	,#0647	
4006 096A;	0024	,#096A	
4008 0C8B;	0025	,#0C8B	
400A 0FAB;	0026	,#0FAB	
400C 12C8;	0027	,#12C8	
400E 15E2;	0028	,#15E2	
4010 18F8;	0029	,#18F8	
4012 1C0B;	0030	,#1C0B	
4014 1F19;	0031	,#1F19	
4016 2223;	0032	,#2223	
4018 2528;	0033	,#2528	
401A 2826;	0034	,#2826	
401C 2B1F;	0035	,#2B1F	
401E 2E11;	0036	,#2E11	
4020 30FB;	0037	,#30FB ..16	
4022 33DE;	0038	,#33DE	
4024 36BA;	0039	,#36BA	
4026 398C;	0040	,#398C	
4028 3C56;	0041	,#3C56	
402A 3F17;	0042	,#3F17	
402C 41CE;	0043	,#41CE	
402E 447A;	0044	,#447A	
4030 471C;	0045	,#471C	



4032 49B4;	0046	,#49B4	
4034 4C3F;	0047	,#4C3F	
4036 4EBF;	0048	,#4EBF	
4038 5133;	0049	,#5133	
403A 539B;	0050	,#539B	
403C 55F5;	0051	,#55F5	
403E 5842;	0052	,#5842	
4040 5A82;	0053	,#5A82	..32
4042 5CB4;	0054	,#5CB4	
4044 5ED7;	0055	,#5ED7	
4046 60EC;	0056	,#60EC	
4048 62F2;	0057	,#62F2	
404A 64E8;	0058	,#64E8	
404C 66CF;	0059	,#66CF	
404E 68A6;	0060	,#68A6	
4050 6A6D;	0061	,#6A6D	
4052 6C24;	0062	,#6C24	
4054 6DCA;	0063	,#6DCA	
4056 6F5F;	0064	,#6F5F	
4058 70E2;	0065	,#70E2	
405A 7255;	0066	,#7255	
405C 73B5;	0067	,#73B5	
405E 7504;	0068	,#7504	
4060 7641;	0069	,#7641	..48
4062 776C;	0070	,#776C	
4064 7884;	0071	,#7884	
4066 798A;	0072	,#798A	
4068 7A7D;	0073	,#7A7D	
406A 7B5D;	0074	,#7B5D	
406C 7C29;	0075	,#7C29	
406E 7CE3;	0076	,#7CE3	
4070 7D8A;	0077	,#7D8A	
4072 7E1D;	0078	,#7E1D	
4074 7E9D;	0079	,#7E9D	
4076 7F09;	0080	,#7F09	
4078 7F62;	0081	,#7F62	
407A 7FA7;	0082	,#7FA7	
407C 7FDB;	0083	,#7FDB	
407E 7FF6;	0084	,#7FF6	
4080 7FFF;	0085	,#7FFF	..64
4082 7FF6;	0086	,#7FF6	
4084 7FD8;	0087	,#7FD8	
4086 7FA7;	0088	,#7FA7	
4088 7F62;	0089	,#7F62	
408A 7F09;	0090	,#7F09	
408C 7E9D;	0091	,#7E9D	
408E 7E1D;	0092	,#7E1D	
4090 7D8A;	0093	,#7D8A	
4092 7CE3;	0094	,#7CE3	
4094 7C29;	0095	,#7C29	
4096 7B5D;	0096	,#7B5D	

ASP FFT Program  
Run Time Code

4098 7A7D;	0097	,#7A7D	
409A 798A;	0098	,#798A	
409C 7884;	0099	,#7884	
409E 776C;	0100	,#776C	
40A0 7641;	0101	,#7641	..80
40A2 7504;	0102	,#7504	
40A4 73B5;	0103	,#73B5	
40A6 7255;	0104	,#7255	
40A8 70E2;	0105	,#70E2	
40AA 6F5F;	0106	,#6F5F	
40AC 6DCA;	0107	,#6DCA	
40AE 6C24;	0108	,#6C24	
40B0 6A6D;	0109	,#6A6D	
40B2 68A6;	0110	,#68A6	
40B4 66CF;	0111	,#66CF	
40B6 64E8;	0112	,#64E8	
40B8 62F2;	0113	,#62F2	
40BA 60EC;	0114	,#60EC	
40BC 5ED7;	0115	,#5ED7	
40BE 5CB4;	0116	,#5CB4	
40C0 5A82;	0117	,#5A82	..96
40C2 5842;	0118	,#5842	
40C4 55F5;	0119	,#55F5	
40C6 539B;	0120	,#539B	
40C8 5133;	0121	,#5133	
40CA 4EBF;	0122	,#4EBF	
40CC 4C3F;	0123	,#4C3F	
40CE 49B4;	0124	,#49B4	
40D0 471C;	0125	,#471C	
40D2 447A;	0126	,#447A	
40D4 41CE;	0127	,#41CE	
40D6 3F17;	0128	,#3F17	
40D8 3C56;	0129	,#3C56	
40DA 398C;	0130	,#398C	
40DC 36BA;	0131	,#36BA	
40DE 33DE;	0132	,#33DE	
40E0 30FB;	0133	,#30FB	..112
40E2 2E11;	0134	,#2E11	
40E4 2B1F;	0135	,#2B1F	
40E6 2826;	0136	,#2826	
40E8 2528;	0137	,#2528	
40EA 2223;	0138	,#2223	
40EC 1F19;	0139	,#1F19	
40EE 1COB;	0140	,#1COB	
40F0 18F8;	0141	,#18F8	
40F2 15E2;	0142	,#15E2	
40F4 12C8;	0143	,#12C8	
40F6 0FAB;	0144	,#0FAB	
40F8 0C8B;	0145	,#0C8B	
40FA 096A;	0146	,#096A	
40FC 0647;	0147	,#0647	

40FE 0324;	0148	,#0324	
4100 0000;	0149	,#0000	..128
4102 FCDC;	0150	,#FCDC	
4104 F9B9;	0151	,#F9B9	
4106 F696;	0152	,#F696	
4108 F375;	0153	,#F375	
410A F055;	0154	,#F055	
410C ED38;	0155	,#ED38	
410E EA1E;	0156	,#EA1E	
4110 E708;	0157	,#E708	
4112 E3F5;	0158	,#E3F5	
4114 E0E7;	0159	,#E0E7	
4116 DDDD;	0160	,#DDDD	
4118 DAD8;	0161	,#DAD8	
411A D7DA;	0162	,#D7DA	
411C D4E1;	0163	,#D4E1	
411E D1EF;	0164	,#D1EF	
4120 CF05;	0165	,#CF05	..144
4122 CC22;	0166	,#CC22	
4124 C946;	0167	,#C946	
4126 C674;	0168	,#C674	
4128 C3AA;	0169	,#C3AA	
412A COE9;	0170	,#COE9	
412C BE32;	0171	,#BE32	
412E BB86;	0172	,#BB86	
4130 B8E4;	0173	,#B8E4	
4132 B64C;	0174	,#B64C	
4134 B3C1;	0175	,#B3C1	
4136 B141;	0176	,#B141	
4138 AECD;	0177	,#AECD	
413A AC65;	0178	,#AC65	
413C AAOB;	0179	,#AAOB	
413E A7BE;	0180	,#A7BE	
4140 A57E;	0181	,#A57E	..160
4142 A34C;	0182	,#A34C	
4144 A129;	0183	,#A129	
4146 9F14;	0184	,#9F14	
4148 9DOE;	0185	,#9DOE	
414A 9B18;	0186	,#9B18	
414C 9931;	0187	,#9931	
414E 975A;	0188	,#975A	
4150 9593;	0189	,#9593	
4152 93DC;	0190	,#93DC	
4154 9236;	0191	,#9236	
4156 90A1;	0192	,#90A1	
4158 8F1E;	0193	,#8F1E	
415A 8DAB;	0194	,#8DAB	
415C 8C4B;	0195	,#8C4B	
415E 8AFC;	0196	,#8AFC	
4160 89BF;	0197	,#89BF	..176
4162 8894;	0198	,#8894	

ASP FFT Program  
Run Time Code

4164 877C;	0199	,#877C	
4166 8676;	0200	,#8676	
4168 8583;	0201	,#8583	
416A 84A3;	0202	,#84A3	
416C 83D7;	0203	,#83D7	
416E 831D;	0204	,#831D	
4170 8276;	0205	,#8276	
4172 81E3;	0206	,#81E3	
4174 8163;	0207	,#8163	
4176 80F7;	0208	,#80F7	
4178 809E;	0209	,#809E	
417A 8059;	0210	,#8059	
417C 8028;	0211	,#8028	
417E 800A;	0212	,#800A	
4180 8000;	0213	,#8000	..192
4182 800A;	0214	,#800A	
4184 8028;	0215	,#8028	
4186 8059;	0216	,#8059	
4188 809E;	0217	,#809E	
418A 80F7;	0218	,#80F7	
418C 8163;	0219	,#8163	
418E 81E3;	0220	,#81E3	
4190 8276;	0221	,#8276	
4192 831D;	0222	,#831D	
4194 83D7;	0223	,#83D7	
4196 84A3;	0224	,#84A3	
4198 8583;	0225	,#8583	
419A 8676;	0226	,#8676	
419C 877C;	0227	,#877C	
419E 8894;	0228	,#8894	
41A0 89BF;	0229	,#89BF	..208
41A2 8AFC;	0230	,#8AFC	
41A4 8C4B;	0231	,#8C4B	
41A6 8DAB;	0232	,#8DAB	
41A8 8F1E;	0233	,#8F1E	
41AA 90A1;	0234	,#90A1	
41AC 9236;	0235	,#9236	
41AE 93DC;	0236	,#93DC	
41B0 9593;	0237	,#9593	
41B2 975A;	0238	,#975A	
41B4 9931;	0239	,#9931	
41B6 9B18;	0240	,#9B18	
41B8 9DOE;	0241	,#9DOE	
41BA 9F14;	0242	,#9F14	
41BC A129;	0243	,#A129	
41BE A34C;	0244	,#A34C	
41C0 A57E;	0245	,#A57E	..224
41C2 A7BE;	0246	,#A7BE	
41C4 AA0B;	0247	,#AA0B	
41C6 AC65;	0248	,#AC65	
41C8 AECD;	0249	,#AECD	



ASP FFT Program  
Run Time Code

41CA B141;	0250	,#B141	
41CC B3C1;	0251	,#B3C1	
41CE B64C;	0252	,#B64C	
41D0 B8E4;	0253	,#B8E4	
41D2 BB86;	0254	,#BB86	
41D4 BE32;	0255	,#BE32	
41D6 COE9;	0256	,#COE9	
41D8 C3AA;	0257	,#C3AA	
41DA C674;	0258	,#C674	
41DC C946;	0259	,#C946	
41DE CC22;	0260	,#CC22	
41E0 CF05;	0261	,#CF05	..240
41E2 D1EF;	0262	,#D1EF	
41E4 D4E1;	0263	,#D4E1	
41E6 D7DA;	0264	,#D7DA	
41E8 DAD8;	0265	,#DAD8	
41EA DDDD;	0266	,#DDDD	
41EC E0E7;	0267	,#E0E7	
41EE E35F;	0268	,#E35F	
41FO E708;	0269	,#E708	
41F2 EA1E;	0270	,#EA1E	
41F4 ED38;	0271	,#ED38	
41F6 F055;	0272	,#F055	
41F8 F375;	0273	,#F375	
41FA F696;	0274	,#F696	
41FC F9B9;	0275	,#F9B9	
41FE FCDC;	0276	,#FCDC	..255
4200 ;	0277 SINLWA:		
4200 ;	0278		



## APPENDIX C

### 1802 INSTRUCTION SET

#### C.1 RCA 1802 COSMAC INSTRUCTION SET

The COSMAC Instruction Summary is given in the tabulations below. Hexadecimal notation is used to refer to the 4-bit binary code. In all registers the bits are numbered from the Least Significant Bit (LSB) to the Most Significant Bit (MSB), starting with 0.

##### Register Operations

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
INCREMENT REG N	INC	1N	$R(N) + 1$
DECREMENT REG N	DEC	2N	$R(N) - 1$
INCREMENT REG X	IRX	60	$R(X) + 1$
GET LOW REG N	GLQ	8N	$R(N).0 \rightarrow D$
PUT LOW REG N	PLO	AN	$D \rightarrow R(N).0$
GET HIGH REG N	GHI	9N	$R(N).1 \rightarrow D$
PUT HIGH REG N	PHI	BN	$D \rightarrow R(N).1$

##### Memory Reference

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
LOAD VIA N	LDN	0N	$M(R(N)) \rightarrow D$ ; FOR N NOT 0
LOAD ADVANCE	LDA	4N	$M(R(N)) \rightarrow D$ ; $R(N) + 1$
LOAD VIA X	LDX	F0	$M(R(X)) \rightarrow D$
LOAD VIA X AND ADVANCE	LDXA	72	$M(R(X)) \rightarrow D$ ; $R(X) + 1$
LOAD IMMEDIATE	LDI	F8	$M(R(P)) \rightarrow D$ ; $R(P) + 1$
STORE VIA N	STR	5N	$D \rightarrow M(R(N))$
STORE VIA X AND DECREMENT	STXD	73	$D \rightarrow M(R(X))$ ; $R(X) - 1$

ASP FFT Program  
1802 Instruction Set

Logic Operations♦♦

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
OR	OR	F1	M(R(X)) OR D→D
OR IMMEDIATE	ORI	F9	M(R(P)) OR D→D; R(P) +1
EXCLUSIVE OR	XOR	F3	M(R(X)) XOR D→D
EXCLUSIVE OR IMMEDIATE	XRI	F8	M(R(P)) XOR D→D; R(P) +1
AND	AND	F2	M(R(X)) AND D→D
AND IMMEDIATE	ANI	FA	M(R(P)) AND D→D; R(P) +1
SHIFT RIGHT	SHR	F6	SHIFT D RIGHT, LSB(D)→DF, 0→MSB(D)
SHIFT RIGHT WITH CARRY	SHRC	76♦	SHIFT D RIGHT, LSB(D)→DF, DF→MSB(D)
RING SHIFT RIGHT	RSHR		
SHIFT LEFT	SHL	FE	SHIFT D LEFT, MSB(D)→DF, 0→LSB(D)
SHIFT LEFT WITH CARRY	SHLC	7E♦	SHIFT D LEFT, MSB(D)→DF, DF→LSB(D)
RING SHIFT LEFT	RSHL		

♦NOTE THIS INSTRUCTION IS ASSOCIATED WITH MORE THAN ONE MNEMONIC. EACH MNEMONIC IS INDIVIDUALLY LISTED.

♦♦NOTE THE ARITHMETIC OPERATIONS AND THE SHIFT INSTRUCTIONS ARE THE ONLY INSTRUCTIONS THAT CAN ALTER THE DF.



### Arithmetic Operations◆◆

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
ADD	ADD	F4	$M(R(X)) + D - DF, D$
ADD IMMEDIATE	ADI	FC	$M(R(P)) + D - DF, D; R(P) + 1$
ADD WITH CARRY	ADC	74	$M(R(X)) + D + DF - DF, D$
ADD WITH CARRY, IMMEDIATE	ADCI	7C	$M(R(P)) + D + DF - DF, D; R(P) + 1$
SUBTRACT D	SD	F5	$M(R(X)) - D - DF, D$
SUBTRACT D IMMEDIATE	SDI	FD	$M(R(P)) - D - DF, D; R(P) + 1$
SUBTRACT D WITH BORROW	SDB	75	$M(R(X)) - D - (NOT DF) - DF, D$
SUBTRACT D WITH BORROW, IMMEDIATE	SDBI	7D	$M(R(P)) - D - (NOT DF) - DF, D; R(P) + 1$
SUBTRACT MEMORY	SM	F7	$D - M(R(X)) - DF, D$
SUBTRACT MEMORY IMMEDIATE	SMI	FF	$D - M(R(P)) - DF, D; R(P) + 1$
SUBTRACT MEMORY WITH BORROW	SMB	77	$D - M(R(X)) - (NOT DF) - DF, D$
SUBTRACT MEMORY WITH BORROW, IMMEDIATE	SMBI	7F	$D - M(R(P)) - (NOT DF) - DF, D; R(P) + 1$

### Branch Instructions — Short Branch

SHORT BRANCH	BR	30	$M(R(P)) \rightarrow R(P).0$
NO SHORT BRANCH (SEE SKP)	NBR	38◆	$R(P) + 1$
SHORT BRANCH IF D=0	BZ	32	IF D=0, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF D NOT 0	BNZ	3A	IF D NOT 0, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF DF=1	BDF	33◆	IF DF=1, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF POS OR ZERO	BPZ		
SHORT BRANCH IF EQUAL OR GREATER	BGE	3B◆	IF DF=0, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF DF=0	BNF		
SHORT BRANCH IF MINUS	BM		
SHORT BRANCH IF LESS	BL		
SHORT BRANCH IF Q=1	BQ	31	IF Q=1, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF Q=0	BNQ	39	IF Q=0, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF EF1=1 (1 = VSS)	B1	34	IF EF1=1, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF EF1=0 (0 = VCC)	BN1	3C	IF EF1=0, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF EF2=1 (1 = VSS)	B2	35	IF EF2=1, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF EF2=0 (0 = VCC)	BN2	3D	IF EF2=0, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF EF3=1 (1 = VSS)	B3	36	IF EF3=1, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF EF3=0 (0 = VCC)	BN3	3E	IF EF3=0, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF EF4=1 (1 = VSS)	B4	37	IF EF4=1, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$
SHORT BRANCH IF EF4=0 (0 = VCC)	BN4	3F	IF EF4=0, $M(R(P)) \rightarrow R(P).0$ ELSE $R(P) + 1$

### Branch Instructions — Long Branch

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
LONG BRANCH	LBR	C0	$M(R(P)) \rightarrow R(P).1$ $M(R(P) + 1) \rightarrow R(P).0$ $R(P) + 2$
NO LONG BRANCH (SEE LSKP)	NLBR	C8♦	
LONG BRANCH IF D=0	LBZ	C2	IF D=0, $M(R(P)) \rightarrow R(P).1$ $M(R(P) + 1) \rightarrow R(P).0$ ELSE $R(P) + 2$
LONG BRANCH IF D NOT 0	LBNZ	CA	IF D NOT 0, $M(R(P)) \rightarrow R(P).1$ $M(R(P) + 1) \rightarrow R(P).0$ ELSE $R(P) + 2$
LONG BRANCH IF DF=1	LBDF	C3	IF DF=1, $M(R(P)) \rightarrow R(P).1$ $M(R(P) + 1) \rightarrow R(P).0$ ELSE $R(P) + 2$
LONG BRANCH IF DF=0	LBNF	CB	IF DF=0, $M(R(P)) \rightarrow R(P).1$ $M(R(P) + 1) \rightarrow R(P).0$ ELSE $R(P) + 2$
LONG BRANCH IF Q=1	LBQ	C1	IF Q=1, $M(R(P)) \rightarrow R(P).1$ $M(R(P) + 1) \rightarrow R(P).0$ ELSE $R(P) + 2$
LONG BRANCH IF Q=0	LBNQ	C9	IF Q=0, $M(R(P)) \rightarrow R(P).1$ $M(R(P) + 1) \rightarrow R(P).0$ ELSE $R(P) + 2$

### Skip Instructions

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
SHORT SKIP (SEE NBR)	SKP	38♦	$R(P) + 1$
LONG SKIP (SEE NLBR)	LSKP	C8♦	$R(P) + 2$
LONG SKIP IF D=0	LSZ	CE	IF D=0, $R(P) + 2$ ELSE CONTINUE
LONG SKIP IF D NOT 0	LSNZ	C6	IF D NOT 0, $R(P) + 2$ ELSE CONTINUE
LONG SKIP IF DF=1	LSDF	CF	IF DF=1, $R(P) + 2$ ELSE CONTINUE
LONG SKIP IF DF=0	LSNF	C7	IF DF=0, $R(P) + 2$ ELSE CONTINUE
LONG SKIP IF Q=1	LSQ	CD	IF Q=1, $R(P) + 2$ ELSE CONTINUE
LONG SKIP IF Q=0	LSNQ	C5	IF Q=0, $R(P) + 2$ ELSE CONTINUE
LONG SKIP IF IE=1	LSIE	CC	IF IE=1, $R(P) + 2$ ELSE CONTINUE

♦NOTE: THIS INSTRUCTION IS ASSOCIATED WITH MORE THAN ONE MNEMONIC. EACH MNEMONIC IS INDIVIDUALLY LISTED.

♦♦NOTE: THE ARITHMETIC OPERATIONS AND THE SHIFT INSTRUCTIONS ARE THE ONLY INSTRUCTIONS THAT CAN ALTER THE DF.

## Control Instructions

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
IDLE	IDL	00	WAIT FOR DMA OR INTERRUPT; $M(R(0)) \rightarrow BUS$
NO OPERATION	NOP	C4	CONTINUE
SET P	SEP	DN	$N \rightarrow P$
SET X	SEX	EN	$N \rightarrow X$
SET Q	SEQ	7B	$1 \rightarrow Q$
RESET Q	REQ	7A	$0 \rightarrow Q$
SAVE	SAV	78	$T \rightarrow M(R(X))$
PUSH X,P TO STACK	MARK	79	$(X,P) \rightarrow T; (X,P) \rightarrow M(R(2))$
RETURN	RET	70	THEN $P \rightarrow X; R(2) - 1$ $M(R(X)) \rightarrow (X,P); R(X) + 1$ $1 \rightarrow IE$
DISABLE	DIS	71	$M(R(X)) \rightarrow (X,P); R(X) + 1$ $0 \rightarrow IE$

## Input—Output Byte Transfer

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
OUTPUT 1	OUT 1	61	$M(R(X)) \rightarrow BUS; R(X) + 1;$ $N \text{ LINES} = 1$
OUTPUT 2	OUT 2	62	$M(R(X)) \rightarrow BUS; R(X) + 1;$ $N \text{ LINES} = 2$
OUTPUT 3	OUT 3	63	$M(R(X)) \rightarrow BUS; R(X) + 1;$ $N \text{ LINES} = 3$
OUTPUT 4	OUT 4	64	$M(R(X)) \rightarrow BUS; R(X) + 1;$ $N \text{ LINES} = 4$
OUTPUT 5	OUT 5	65	$M(R(X)) \rightarrow BUS; R(X) + 1;$ $N \text{ LINES} = 5$
OUTPUT 6	OUT 6	66	$M(R(X)) \rightarrow BUS; R(X) + 1;$ $N \text{ LINES} = 6$
OUTPUT 7	OUT 7	67	$M(R(X)) \rightarrow BUS; R(X) + 1;$ $N \text{ LINES} = 7$
INPUT 1	INP 1	69	$BUS \rightarrow M(R(X)); BUS \rightarrow D;$ $N \text{ LINES} = 1$
INPUT 2	INP 2	6A	$BUS \rightarrow M(R(X)); BUS \rightarrow D;$ $N \text{ LINES} = 2$
INPUT 3	INP 3	6B	$BUS \rightarrow M(R(X)); BUS \rightarrow D;$ $N \text{ LINES} = 3$
INPUT 4	INP 4	6C	$BUS \rightarrow M(R(X)); BUS \rightarrow D;$ $N \text{ LINES} = 4$
INPUT 5	INP 5	6D	$BUS \rightarrow M(R(X)); BUS \rightarrow D;$ $N \text{ LINES} = 5$
INPUT 6	INP 6	6E	$BUS \rightarrow M(R(X)); BUS \rightarrow D;$ $N \text{ LINES} = 6$
INPUT 7	INP 7	6F	$BUS \rightarrow M(R(X)); BUS \rightarrow D;$ $N \text{ LINES} = 7$

◆NOTE THIS INSTRUCTION IS ASSOCIATED WITH MORE THAN ONE MNEMONIC. EACH MNEMONIC IS INDIVIDUALLY LISTED.  
◆◆NOTE THE ARITHMETIC OPERATIONS AND THE SHIFT INSTRUCTIONS ARE THE ONLY INSTRUCTIONS THAT CAN ALTER THE OF





## APPENDIX D

### PROGRAM VALIDATION OUTPUT

The following listings are examples of the Program Validation Outputs. The first section gives snapshots of the outputs of the subprograms (INPUT, SCALE, FFT, etc.), first for a single P-FFT, then for a single S-FFT. The second section shows a sequence of inputs (from Impulse to DC) and the resultant outputs, for both a P-FFT and an S-FFT.

All the samples included are for an FFT Length of 128 (other runs were used to confirm operation of all various FFT lengths). Data is utilized as 16-bit Fixed Point Twos-Complement, represented in hexadecimal notation (thus four symbols per data item). Data in locations 3601 to 3A00 IS P-FFT data. In locations 3A01 to 3E00 the data is either in-place S-FFT data, or P-FFT data which has been moved to the S-FIFO for computational efficiency (see Section 5.1.3.7).

For all of the outputs listed the SHAPE subprogram was not called. This is because the time domain lengths utilized are assumed to be perfect segments of a periodic waveform. Thus Frequency Leakage is not an issue. Said another way, for these examples, pretty outputs were desired. Other Program Validation sequences were run where the SHAPE subprogram was tested.

# ASP FFT Program Program Validation Output

Finally, for most of these examples, the MAGAPX subprogram was not used. In its place a MAGSQ (Magnitude Squared) routine was utilized (program listing not available). This simply performed the Sum of the Squares ( $R^2 + I^2$ ) on the Complex data. Again, these outputs are included for cosmetic purposes. Other listings validated the performance of the MAGAPX subprogram.

## D.1 SUBPROGRAM VALIDATION SERIES

### D.1.1 P-FFT, Impulse Input

Input Series

```

3601 7FFF 0000 0000 0000 0000 0000 0000 0000;
3611 0000 0000 0000 0000 0000 0000 0000 0000;
3621 0000 0000 0000 0000 0000 0000 0000 0000;
3631 0000 0000 0000 0000 0000 0000 0000 0000;
3641 0000 0000 0000 0000 0000 0000 0000 0000;
3651 0000 0000 0000 0000 0000 0000 0000 0000;
3661 0000 0000 0000 0000 0000 0000 0000 0000;
3671 0000 0000 0000 0000 0000 0000 0000 0000;
3681 0000 0000 0000 0000 0000 0000 0000 0000;
3691 0000 0000 0000 0000 0000 0000 0000 0000;
36A1 0000 0000 0000 0000 0000 0000 0000 0000;
36B1 0000 0000 0000 0000 0000 0000 0000 0000;
36C1 0000 0000 0000 0000 0000 0000 0000 0000;
36D1 0000 0000 0000 0000 0000 0000 0000 0000;
36E1 0000 0000 0000 0000 0000 0000 0000 0000;
36F1 0000 0000 0000 0000 0000 0000 0000 0000

```

SCALE Output

```

3601 7FFF 0000 0000 0000 0000 0000 0000 0000;
3611 0000 0000 0000 0000 0000 0000 0000 0000;
3621 0000 0000 0000 0000 0000 0000 0000 0000;
3631 0000 0000 0000 0000 0000 0000 0000 0000;
3641 0000 0000 0000 0000 0000 0000 0000 0000;
3651 0000 0000 0000 0000 0000 0000 0000 0000;
3661 0000 0000 0000 0000 0000 0000 0000 0000;
3671 0000 0000 0000 0000 0000 0000 0000 0000;
3681 0000 0000 0000 0000 0000 0000 0000 0000;
3691 0000 0000 0000 0000 0000 0000 0000 0000;
36A1 0000 0000 0000 0000 0000 0000 0000 0000;
36B1 0000 0000 0000 0000 0000 0000 0000 0000;
36C1 0000 0000 0000 0000 0000 0000 0000 0000;
36D1 0000 0000 0000 0000 0000 0000 0000 0000;
36E1 0000 0000 0000 0000 0000 0000 0000 0000;
36F1 0000 0000 0000 0000 0000 0000 0000 0000

```

ASP FFT Program  
Program Validation Output

FFT Output

```

3A01 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A11 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A21 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A31 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A41 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A51 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A61 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A71 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A81 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A91 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3AA1 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3AB1 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3AC1 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3AD1 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3AE1 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3AF1 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000

```

UNSCRM Output

```

3A01 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A11 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A21 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A31 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A41 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A51 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A61 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A71 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A81 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3A91 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3AA1 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3AB1 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3AC1 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3AD1 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3AE1 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000;
3AF1 1FFF 0000 1FFF 0000 1FFF 0000 1FFF 0000

```

ASP FFT Program  
Program Validation Output

ORDER Output

```

3A01 3FFE 0000 3FFE 0000 3FFE 0000 3FFE 0000;
3A11 3FFE 0000 3FFE 0000 3FFE 0000 3FFE 0000;
3A21 3FFE 0000 3FFE 0000 3FFE 0000 3FFE 0000;
3A31 3FFE 0000 3FFE 0000 3FFE 0000 3FFE 0000;
3A41 3FFE 0000 3FFE 0000 3FFE 0000 3FFE 0000;
3A51 3FFE 0000 3FFE 0000 3FFE 0000 3FFE 0000;
3A61 3FFE 0000 3FFE 0000 3FFE 0000 3FFE 0000;
3A71 3FFE 0000 3FFE 0000 3FFE 0000 3FFE 0000;
3A81 3FFE 0000 3FFE 0000 3FFE 0000 3FFE 0000;
3A91 3FFE 0000 3FFE 0000 3FFE 0000 3FFE 0000;
3AA1 3FFE 0000 3FFE 0000 3FFE 0000 3FFE 0000;
3AB1 3FFE 0000 3FFE 0000 3FFE 0000 3FFE 0000;
3AC1 3FFE 0000 3FFE 0000 3FFE 0000 3FFE 0000;
3AD1 3FFE 0000 3FFE 0000 3FFE 0000 3FFE 0000;
3AE1 3FFE 0000 3FFE 0000 3FFE 0000 3FFE 0000;
3AF1 3FFE 0000 3FFE 0000 3FFE 0000 3FFE 0000

```

MAGSQ Output

```

3A01 7FFC 0000 7FFC 0000 7FFC 0000 7FFC 0000;
3A11 7FFC 0000 7FFC 0000 7FFC 0000 7FFC 0000;
3A21 7FFC 0000 7FFC 0000 7FFC 0000 7FFC 0000;
3A31 7FFC 0000 7FFC 0000 7FFC 0000 7FFC 0000;
3A41 7FFC 0000 7FFC 0000 7FFC 0000 7FFC 0000;
3A51 7FFC 0000 7FFC 0000 7FFC 0000 7FFC 0000;
3A61 7FFC 0000 7FFC 0000 7FFC 0000 7FFC 0000;
3A71 7FFC 0000 7FFC 0000 7FFC 0000 7FFC 0000;
3A81 7FFC 0000 7FFC 0000 7FFC 0000 7FFC 0000;
3A91 7FFC 0000 7FFC 0000 7FFC 0000 7FFC 0000;
3AA1 7FFC 0000 7FFC 0000 7FFC 0000 7FFC 0000;
3AB1 7FFC 0000 7FFC 0000 7FFC 0000 7FFC 0000;
3AC1 7FFC 0000 7FFC 0000 7FFC 0000 7FFC 0000;
3AD1 7FFC 0000 7FFC 0000 7FFC 0000 7FFC 0000;
3AE1 7FFC 0000 7FFC 0000 7FFC 0000 7FFC 0000;
3AF1 7FFC 0000 7FFC 0000 7FFC 0000 7FFC 0000

```



D.1.2 S-FFT, DC Input

Input Series

3A01	0800	0800	0800	0800	0800	0800	0800	0800;
3A11	0800	0800	0800	0800	0800	0800	0800	0800;
3A21	0800	0800	0800	0800	0800	0800	0800	0800;
3A31	0800	0800	0800	0800	0800	0800	0800	0800;
3A41	0800	0800	0800	0800	0800	0800	0800	0800;
3A51	0800	0800	0800	0800	0800	0800	0800	0800;
3A61	0800	0800	0800	0800	0800	0800	0800	0800;
3A71	0800	0800	0800	0800	0800	0800	0800	0800;
3A81	0800	0800	0800	0800	0800	0800	0800	0800;
3A91	0800	0800	0800	0800	0800	0800	0800	0800;
3AA1	0800	0800	0800	0800	0800	0800	0800	0800;
3AB1	0800	0800	0800	0800	0800	0800	0800	0800;
3AC1	0800	0800	0800	0800	0800	0800	0800	0800;
3AD1	0800	0800	0800	0800	0800	0800	0800	0800;
3AE1	0800	0800	0800	0800	0800	0800	0800	0800;
3AF1	0800	0800	0800	0800	0800	0800	0800	0800;

SCALE Output

3A01	4000	4000	4000	4000	4000	4000	4000	4000;
3A11	4000	4000	4000	4000	4000	4000	4000	4000;
3A21	4000	4000	4000	4000	4000	4000	4000	4000;
3A31	4000	4000	4000	4000	4000	4000	4000	4000;
3A41	4000	4000	4000	4000	4000	4000	4000	4000;
3A51	4000	4000	4000	4000	4000	4000	4000	4000;
3A61	4000	4000	4000	4000	4000	4000	4000	4000;
3A71	4000	4000	4000	4000	4000	4000	4000	4000;
3A81	4000	4000	4000	4000	4000	4000	4000	4000;
3A91	4000	4000	4000	4000	4000	4000	4000	4000;
3AA1	4000	4000	4000	4000	4000	4000	4000	4000;
3AB1	4000	4000	4000	4000	4000	4000	4000	4000;
3AC1	4000	4000	4000	4000	4000	4000	4000	4000;
3AD1	4000	4000	4000	4000	4000	4000	4000	4000;
3AE1	4000	4000	4000	4000	4000	4000	4000	4000;
3AF1	4000	4000	4000	4000	4000	4000	4000	4000;

ASP FFT Program  
Program Validation Output

FFT Output

```

3A01 38EE 394E FF02 0082 FF80 0000 0080 FE00;
3A11 FF58 0110 FEA8 FEF0 FFEE 005A 0212 FFA6;
3A21 0005 018D FE91 FFDD FFDB 0007 018F FEFB;
3A31 FF32 0068 FF62 FE2C FF96 00A0 01D6 00CC;
3A41 060E FD0D FF32 00C9 009A 000D 00F6 FF4D;
3A51 00A5 004A FF2D FF46 0075 00E0 00E5 0264;
3A61 00EC 00C2 FFB4 FF82 00B6 0066 017A 007E;
3A71 006A 000B 0002 FEE7 FFDA 0115 FCEA 06C9;
3A81 0037 F77F 005B 004F 0017 FF81 014F FEA1;
3A91 FFBF FFD6 FF47 FEF2 0063 0044 0297 0158;
3AA1 004E FF78 FF06 FFC4 006A FFCA 0242 FFA2;
3AB1 FFBF FF97 0023 FE6F FFF3 0091 022F 0469;
3AC1 0399 021C FE91 0090 0027 007A 00AB FEDA;
3AD1 0004 0111 FED8 FED1 FFD4 0115 FFFC 0109;
3AE1 00DA 01DB FE62 FFAB 0030 00C1 00F4 FFD9;
3AF1 FFF2 0091 FFC2 FE79 FF2A 0115 F716 01E1

```

UNSCRM Output

```

3A01 38EE 394E 0037 F77F 060E FD0D 0399 021C;
3A11 0005 018D 004E FF78 00EC 00C2 00DA 01BB;
3A21 FF58 0110 FFBF FFD6 00A5 004A 0004 0111;
3A31 FF32 0068 FFBF FF97 006A 000B FFF2 0091;
3A41 FF80 0000 0017 FF81 009A 000D 0027 007A;
3A51 FFDB 0007 006A FFCA 00B6 0066 0030 00C1;
3A61 FFEE 005A 0063 0044 0075 00E0 FFD4 0115;
3A71 FF96 00A0 FFF3 0091 FFD4 0115 FF2A 0115;
3A81 FF02 0082 005B 004F FF32 00C9 FE91 0090;
3A91 FE91 FFDD FF06 FFC4 FFB4 FF82 FE62 FFA8;
3AA1 FEA8 FEF0 FF47 FEF2 FF2D FF46 FED8 FED1;
3AB1 FF62 FE2C 0023 FE6F 0002 FEE7 FFC2 FE79;
3AC1 0080 FE00 014F FEA1 00F6 FF4D 00AB FEDA;
3AD1 018F FEFB 0242 FFA2 017A 007E 00F4 FFD9;
3AE1 0212 FFA6 0297 0158 00E5 0264 FFFC 0109;
3AF1 01D6 00CC 022F 0469 FCEA 06C9 F716 01E1

```

ASP FFT Program  
Program Validation Output

ORDER Output

```

3A01 725C 0000 F81E F66B 02F0 F666 0601 FDAE;
3A11 0241 010C 005B FF04 0268 FEB9 0376 007F;
3A21 018F 01D7 0077 009F 019B 0018 0203 0183;
3A31 0098 0233 FFDA 0136 009C 00CD 00C8 01C0;
3A41 FFA6 0212 FF1E 012A FFBE 00CD FFD0 0178;
3A51 FEEC 018F FE97 00B1 FF31 005A FF41 00D9;
3A61 FE91 00CD FE6E 0001 FFC8 0067 FF25 0028;
3A71 FE9F 000E FEA7 FF5A FF4A FF32 0062 FFB8;
3A81 FF02 FF80 FF22 FEF5 FFC2 FEE6 FFD8 FF5A;
3A91 FF87 FF4A FFB5 FED8 0060 FF09 0056 FF67;
3AA1 0005 FF63 0035 FF0B 00B1 FF3A 00AB FFB7;
3AB1 0050 FFB5 0078 FF6C 00DE FFA7 00BA 0022;
3AC1 005A 0012 0078 FFD0 00C4 000F 008E 007A;
3AD1 0028 0059 0043 0015 0083 004C 003B 00A1;
3AE1 FFD8 006D FFFA 001D FF68 005B FFEF 0094;
3AF1 FF99 004C FFC5 FFFA 0008 0022 FF2E 00CD

```

MAGAPX Output

```

3A01 392E 0000 06B0 0000 04CC 0000 032E 0000;
3A11 0142 0000 0082 0000 0168 0000 018B 0000;
3A21 014C 0000 0068 0000 00CD 0000 0158 0000;
3A31 0110 0000 009B 0000 0088 0000 00F8 0000;
3A41 0109 0000 00C8 0000 0066 0000 00BC 0000;
3A51 0102 0000 00CE 0000 0072 0000 009C 0000;
3A61 00DA 0000 00C9 0000 003C 0000 006E 0000;
3A71 00B1 0000 00C4 0000 0094 0000 0040 0000;
3A81 0092 0000 00BA 0000 008D 0000 0053 0000;
3A91 0074 0000 008E 0000 0082 0000 005A 0000;
3AA1 004F 0000 007B 0000 008E 0000 005C 0000;
3AB1 003A 0000 0066 0000 0076 0000 005D 0000;
3AC1 002D 0000 0040 0000 0062 0000 0064 0000;
3AD1 0030 0000 0020 0000 004E 0000 0052 0000;
3AE1 0036 0000 000E 0000 005C 0000 004A 0000;
3AF1 0044 0000 001E 0000 0011 0000 009E 0000

```

ASP FFT Program  
Program Validation Output

D.2 INPUT/OUTPUT VALIDATION SERIES

D.2.1 P-FFT, Impulse to DC Series

Impulse Time Input  
(1 of 128)

3601	0000	0000	0000	0000	0000	0000	0000	0000;
3611	0000	0000	0000	0000	0000	0000	0000	0000;
3621	0000	0000	0000	0000	0000	0000	0000	0000;
3631	0000	0000	0000	0000	0000	0000	0000	0000;
3641	0000	0000	0000	0000	0000	0000	0000	0000;
3651	0000	0000	0000	0000	0000	0000	0000	0000;
3661	0000	0000	0000	0000	0000	0000	0000	0000;
3671	0000	0000	0000	0000	0000	0000	0000	0000;
3681	0000	0000	0000	0000	0000	0000	0000	0000;
3691	0000	0000	0000	0000	0000	0000	0000	0000;
36A1	0000	0000	0000	0000	0000	0000	0000	0000;
36B1	0000	0000	0000	0000	0000	0000	0000	0000;
36C1	0000	0000	0000	0000	0000	0000	0000	0000;
36D1	0000	0000	0000	0000	0000	0000	0000	0000;
36E1	0000	0000	0000	0000	0000	0000	0000	0000;
36F1	0000	0000	0000	0000	0000	0000	0000	0000

Frequency Output

3A01	2000	0000	2000	0000	2000	0000	2000	0000;
3A11	2000	0000	2000	0000	2000	0000	2000	0000;
3A21	2000	0000	2000	0000	2000	0000	2000	0000;
3A31	2000	0000	2000	0000	2000	0000	2000	0000;
3A41	2000	0000	2000	0000	2000	0000	2000	0000;
3A51	2000	0000	2000	0000	2000	0000	2000	0000;
3A61	2000	0000	2000	0000	2000	0000	2000	0000;
3A71	2000	0000	2000	0000	2000	0000	2000	0000;
3A81	2000	0000	2000	0000	2000	0000	2000	0000;
3A91	2000	0000	2000	0000	2000	0000	2000	0000;
3AA1	2000	0000	2000	0000	2000	0000	2000	0000;
3AB1	2000	0000	2000	0000	2000	0000	2000	0000;
3AC1	2000	0000	2000	0000	2000	0000	2000	0000;
3AD1	2000	0000	2000	0000	2000	0000	2000	0000;
3AE1	2000	0000	2000	0000	2000	0000	2000	0000;
3AF1	2000	0000	2000	0000	2000	0000	2000	0000



(2 of 128) Time Input

```

3601 0000 0000 0000 0000 0000 0000 0000 0000;
3611 0000 0000 0000 0000 0000 0000 0000 0000;
3621 0000 0000 0000 0000 0000 0000 0000 0000;
3631 0000 0000 0000 0000 0000 0000 0000 0000;
3641 0000 0000 0000 0000 0000 0000 0000 0000;
3651 0000 0000 0000 0000 0000 0000 0000 0000;
3661 0000 0000 0000 0000 0000 0000 0000 0000;
3671 0000 0000 0000 0000 0000 0000 0000 0000;
3681 0000 0000 0000 0000 0000 0000 0000 0000;
3691 0000 0000 0000 0000 0000 0000 0000 0000;
36A1 0000 0000 0000 0000 0000 0000 0000 0000;
36B1 0000 0000 0000 0000 0000 0000 0000 0000;
36C1 0000 0000 0000 0000 0000 0000 0000 0000;
36D1 0000 0000 0000 0000 0000 0000 0000 0000;
36E1 0000 0000 0000 0000 0000 0000 0000 0000;
36F1 0000 0000 0000 0000 0000 0000 0000 0000

```

Frequency Output

```

3A01 2000 0000 13FE 0000 13FF 0000 13F5 0000;
3A11 13EC 0000 13E1 0000 13D4 0000 13C4 0000;
3A21 13B2 0000 13A0 0000 1387 0000 136F 0000;
3A31 1357 0000 1336 0000 131F 0000 12F6 0000;
3A41 12D4 0000 12B0 0000 1289 0000 1262 0000;
3A51 1239 0000 120F 0000 11E2 0000 11C0 0000;
3A61 1187 0000 1159 0000 1129 0000 10F9 0000;
3A71 10C7 0000 1096 0000 1064 0000 1032 0000;
3A81 1000 0000 0F28 0000 0F04 0000 0ED6 0000;
3A91 0EA8 0000 0E7A 0000 0E4C 0000 0E1A 0000;
3AA1 0DEA 0000 0DB0 0000 0E1E 0000 0DF1 0000;
3AB1 0DC7 0000 0D9E 0000 0D77 0000 0D50 0000;
3AC1 0D2C 0000 0D0A 0000 0CE1 0000 0CCA 0000;
3AD1 0CA9 0000 0C91 0000 0C79 0000 0C60 0000;
3AE1 0C4E 0000 0C3C 0000 0C2C 0000 0C1F 0000;
3AF1 0C14 0000 0C0B 0000 0C01 0000 0C02 0000

```

ASP FFT Program  
Program Validation Output

(4 of 128) Time Input

```

3601 0000 0000 0000 0000 0000 0000 0000 0000;
3611 0000 0000 0000 0000 0000 0000 0000 0000;
3621 0000 0000 0000 0000 0000 0000 0000 0000;
3631 0000 0000 0000 0000 0000 0000 0000 0000;
3641 0000 0000 0000 0000 0000 0000 0000 0000;
3651 0000 0000 0000 0000 0000 0000 0000 0000;
3661 0000 0000 0000 0000 0000 0000 0000 0000;
3671 0000 0000 0000 0000 0000 0000 0000 0000;
3681 0000 0000 0000 0000 0000 0000 0000 0000;
3691 0000 0000 0000 0000 0000 0000 0000 0000;
36A1 0000 0000 0000 0000 0000 0000 0000 0000;
36B1 0000 0000 0000 0000 0000 0000 0000 0000;
36C1 0000 0000 0000 0000 0000 0000 0000 0000;
36D1 0000 0000 0000 0000 0000 0000 0000 0000;
36E1 0000 0000 0000 0000 0000 0000 0000 0000;
36F1 0000 0000 0000 0000 0000 0000 0000 0000

```

Frequency Output

```

3A01 204A 0000 036F 0000 013C 0000 0122 0000;
3A11 00CE 0000 00BC 0000 00B2 0000 00A2 0000;
3A21 0092 0000 0098 0000 0082 0000 0086 0000;
3A31 0086 0000 007E 0000 0082 0000 007E 0000;
3A41 0080 0000 0078 0000 0086 0000 007A 0000;
3A51 0082 0000 0081 0000 007E 0000 0080 0000;
3A61 0082 0000 007D 0000 0085 0000 007C 0000;
3A71 0080 0000 0082 0000 0082 0000 007A 0000;
3A81 0086 0000 007A 0000 0082 0000 007E 0000;
3A91 0080 0000 007C 0000 0085 0000 0079 0000;
3AA1 0082 0000 007C 0000 007E 0000 007D 0000;
3AB1 0082 0000 007A 0000 0086 0000 0078 0000;
3AC1 0080 0000 007E 0000 0080 0000 007E 0000;
3AD1 0086 0000 008E 0000 008C 0000 00A2 0000;
3AE1 0098 0000 00A8 0000 00B8 0000 00C8 0000;
3AF1 00D8 0000 0128 0000 014A 0000 037F 0000

```

(8 of 128) Time Input

```

3601 0800 0800 0800 0800 0800 0800 0800 0800;
3611 0000 0000 0000 0000 0000 0000 0000 0000;
3621 0000 0000 0000 0000 0000 0000 0000 0000;
3631 0000 0000 0000 0000 0000 0000 0000 0000;
3641 0000 0000 0000 0000 0000 0000 0000 0000;
3651 0000 0000 0000 0000 0000 0000 0000 0000;
3661 0000 0000 0000 0000 0000 0000 0000 0000;
3671 0000 0000 0000 0000 0000 0000 0000 0000;
3681 0000 0000 0000 0000 0000 0000 0000 0000;
3691 0000 0000 0000 0000 0000 0000 0000 0000;
36A1 0000 0000 0000 0000 0000 0000 0000 0000;
36B1 0000 0000 0000 0000 0000 0000 0000 0000;
36C1 0000 0000 0000 0000 0000 0000 0000 0000;
36D1 0000 0000 0000 0000 0000 0000 0000 0000;
36E1 0000 0000 0000 0000 0000 0000 0000 0000;
36F1 0000 0000 0000 0000 0000 0000 0000 0000

```

Frequency Output

```

3A01 250C 0000 153D 0000 1481 0000 1390 0000;
3A11 12D7 0000 124D 0000 11FA 0000 11A0 0000;
3A21 113A 0000 10ED 0000 10A4 0000 1069 0000;
3A31 1023 0000 0FF0 0000 0FB0 0000 0F85 0000;
3A41 0F50 0000 0F25 0000 0EF7 0000 0ED6 0000;
3A51 0EB5 0000 0E99 0000 0E94 0000 0E27 0000;
3A61 0E12 0000 0DDA 0000 0DAA 0000 0D9B 0000;
3A71 0D99 0000 0D78 0000 0D49 0000 0D19 0000;
3A81 0D64 0000 0D15 0000 0D49 0000 0D78 0000;
3A91 0D45 0000 0D97 0000 0DAE 0000 0DDE 0000;
3AA1 0E0E 0000 0E23 0000 0E90 0000 0E9D 0000;
3AB1 0EB5 0000 0ECA 0000 0EF7 0000 0F29 0000;
3AC1 0F50 0000 0F7D 0000 0FB5 0000 0FEC 0000;
3AD1 101B 0000 1061 0000 10A8 0000 10ED 0000;
3AE1 1136 0000 1194 0000 11EE 0000 1249 0000;
3AF1 12D3 0000 137C 0000 147D 0000 1535 0000

```

ASP FFT Program  
Program Validation Output

(16 of 128) Time Input

```

3601 0000 0000 0000 0000 0000 0000 0000 0000;
3611 0000 0000 0000 0000 0000 0000 0000 0000;
3621 0000 0000 0000 0000 0000 0000 0000 0000;
3631 0000 0000 0000 0000 0000 0000 0000 0000;
3641 0000 0000 0000 0000 0000 0000 0000 0000;
3651 0000 0000 0000 0000 0000 0000 0000 0000;
3661 0000 0000 0000 0000 0000 0000 0000 0000;
3671 0000 0000 0000 0000 0000 0000 0000 0000;
3681 0000 0000 0000 0000 0000 0000 0000 0000;
3691 0000 0000 0000 0000 0000 0000 0000 0000;
36A1 0000 0000 0000 0000 0000 0000 0000 0000;
36B1 0000 0000 0000 0000 0000 0000 0000 0000;
36C1 0000 0000 0000 0000 0000 0000 0000 0000;
36D1 0000 0000 0000 0000 0000 0000 0000 0000;
36E1 0000 0000 0000 0000 0000 0000 0000 0000;
36F1 0000 0000 0000 0000 0000 0000 0000 0000

```

Frequency Output

```

3A01 203A 0000 0372 0000 0140 0000 0122 0000;
3A11 00CD 0000 00BF 0000 00B2 0000 00A3 0000;
3A21 0092 0000 009B 0000 0088 0000 008A 0000;
3A31 0086 0000 007B 0000 0082 0000 007F 0000;
3A41 007C 0000 0075 0000 0082 0000 0077 0000;
3A51 007E 0000 0078 0000 0078 0000 007B 0000;
3A61 007E 0000 0077 0000 0080 0000 007D 0000;
3A71 007F 0000 007E 0000 0082 0000 0078 0000;
3A81 0086 0000 0078 0000 007E 0000 007E 0000;
3A91 007B 0000 0075 0000 0080 0000 0073 0000;
3AA1 007E 0000 0077 0000 0078 0000 0078 0000;
3AB1 007E 0000 0073 0000 0082 0000 0071 0000;
3AC1 007C 0000 0077 0000 007A 0000 007B 0000;
3AD1 0082 0000 0082 0000 0080 0000 0097 0000;
3AE1 0092 0000 009F 0000 00AE 0000 00BB 0000;
3AF1 00CD 0000 011E 0000 0140 0000 0372 0000

```



(32 of 128) Time Input

3601	0800	0800	0800	0800	0800	0800	0800	0800
3611	0800	0800	0800	0800	0800	0800	0800	0800
3621	0800	0800	0800	0800	0800	0800	0800	0800
3631	0800	0800	0800	0800	0800	0800	0800	0800
3641	0000	0000	0000	0000	0000	0000	0000	0000
3651	0000	0000	0000	0000	0000	0000	0000	0000
3661	0000	0000	0000	0000	0000	0000	0000	0000
3671	0000	0000	0000	0000	0000	0000	0000	0000
3681	0000	0000	0000	0000	0000	0000	0000	0000
3691	0000	0000	0000	0000	0000	0000	0000	0000
36A1	0000	0000	0000	0000	0000	0000	0000	0000
36B1	0000	0000	0000	0000	0000	0000	0000	0000
36C1	0000	0000	0000	0000	0000	0000	0000	0000
36D1	0000	0000	0000	0000	0000	0000	0000	0000
36E1	0000	0000	0000	0000	0000	0000	0000	0000
36F1	0000	0000	0000	0000	0000	0000	0000	0000

Frequency Output

3A01	2486	0000	1535	0000	1486	0000	137E	0000
3A11	12D9	0000	124B	0000	11F6	0000	1192	0000
3A21	1138	0000	10ED	0000	10A4	0000	105B	0000
3A31	100F	0000	0FE9	0000	0FB1	0000	0F75	0000
3A41	0F4E	0000	0F21	0000	0EF7	0000	0EC7	0000
3A51	0EAD	0000	0E95	0000	0E8C	0000	0E1F	0000
3A61	0E10	0000	0DD6	0000	0DAC	0000	0D95	0000
3A71	0D9B	0000	0D6E	0000	0D38	0000	0D0F	0000
3A81	0D66	0000	0D0B	0000	0D38	0000	0D72	0000
3A91	0D97	0000	0D95	0000	0DB0	0000	0DDA	0000
3AA1	0E0C	0000	0E1B	0000	0E8C	0000	0E95	0000
3AB1	0EAD	0000	0EC3	0000	0EF7	0000	0F21	0000
3AC1	0F4E	0000	0F6D	0000	0FA9	0000	0FED	0000
3AD1	1007	0000	1057	0000	10A8	0000	10ED	0000
3AE1	1134	0000	118A	0000	11EA	0000	1247	0000
3AF1	12D5	0000	136A	0000	1482	0000	152D	0000

ASP FFT Program  
Program Validation Output

(64 of 128) Time Input

3601	0800	0800	0800	0800	0800	0800	0800	0800;
3611	0800	0800	0800	0800	0800	0800	0800	0800;
3621	0800	0800	0800	0800	0800	0800	0800	0800;
3631	0800	0800	0800	0800	0800	0800	0800	0800;
3641	0800	0800	0800	0800	0800	0800	0800	0800;
3651	0800	0800	0800	0800	0800	0800	0800	0800;
3661	0800	0800	0800	0800	0800	0800	0800	0800;
3671	0800	0800	0800	0800	0800	0800	0800	0800;
3681	0000	0000	0000	0000	0000	0000	0000	0000;
3691	0000	0000	0000	0000	0000	0000	0000	0000;
36A1	0000	0000	0000	0000	0000	0000	0000	0000;
36B1	0000	0000	0000	0000	0000	0000	0000	0000;
36C1	0000	0000	0000	0000	0000	0000	0000	0000;
36D1	0000	0000	0000	0000	0000	0000	0000	0000;
36E1	0000	0000	0000	0000	0000	0000	0000	0000;
36F1	0000	0000	0000	0000	0000	0000	0000	0000;

Frequency Output

3A01	202A	0000	036E	0000	013F	0000	0120	0000;
3A11	00CA	0000	00BC	0000	00AF	0000	009F	0000;
3A21	0090	0000	0096	0000	0083	0000	0086	0000;
3A31	0085	0000	0077	0000	007D	0000	0079	0000;
3A41	0078	0000	006B	0000	0081	0000	0073	0000;
3A51	007D	0000	0074	0000	0075	0000	0074	0000;
3A61	007C	0000	0073	0000	007B	0000	0076	0000;
3A71	007A	0000	007E	0000	007F	0000	0076	0000;
3A81	0086	0000	0072	0000	007B	0000	007A	0000;
3A91	007A	0000	0072	0000	007B	0000	0073	0000;
3AA1	007C	0000	0074	0000	0071	0000	0074	0000;
3AB1	0079	0000	006F	0000	007D	0000	006B	0000;
3AC1	0078	0000	0075	0000	0079	0000	0077	0000;
3AD1	0081	0000	007E	0000	007D	0000	0096	0000;
3AE1	0090	0000	009B	0000	00A7	0000	00B8	0000;
3AF1	00C6	0000	011C	0000	013B	0000	036E	0000;

ASP FFT Program  
Program Validation Output

DC Time Input

```

3601 0800 0800 0800 0800 0800 0800 0800 0800;
3611 0800 0800 0800 0800 0800 0800 0800 0800;
3621 0800 0800 0800 0800 0800 0800 0800 0800;
3631 0800 0800 0800 0800 0800 0800 0800 0800;
3641 0800 0800 0800 0800 0800 0800 0800 0800;
3651 0800 0800 0800 0800 0800 0800 0800 0800;
3661 0800 0800 0800 0800 0800 0800 0800 0800;
3671 0800 0800 0800 0800 0800 0800 0800 0800;
3681 0800 0800 0800 0800 0800 0800 0800 0800;
3691 0800 0800 0800 0800 0800 0800 0800 0800;
36A1 0800 0800 0800 0800 0800 0800 0800 0800;
36B1 0800 0800 0800 0800 0800 0800 0800 0800;
36C1 0800 0800 0800 0800 0800 0800 0800 0800;
36D1 0800 0800 0800 0800 0800 0800 0800 0800;
36E1 0800 0800 0800 0800 0800 0800 0800 0800;
36F1 0800 0800 0800 0800 0800 0800 0800 0800;

```

Frequency Output

```

3A01 2402 0000 1530 0000 148A 0000 1374 0000;
3A11 12CE 0000 1245 0000 11ED 0000 118D 0000;
3A21 112E 0000 10E6 0000 10A2 0000 1053 0000;
3A31 100D 0000 0FE6 0000 0FAF 0000 0F6E 0000;
3A41 0F52 0000 0F12 0000 0EF1 0000 0EBC 0000;
3A51 0EAB 0000 0E8D 0000 0E88 0000 0E16 0000;
3A61 0E0E 0000 0DCF 0000 0DA7 0000 0D8F 0000;
3A71 0D98 0000 0D68 0000 0D34 0000 0D06 0000;
3A81 0D5A 0000 0D06 0000 0D34 0000 0D6C 0000;
3A91 0D98 0000 0D8B 0000 0DA7 0000 0DCF 0000;
3AA1 0E0A 0000 0E12 0000 0E88 0000 0E8D 0000;
3AB1 0EAB 0000 0EB8 0000 0EF1 0000 0F16 0000;
3AC1 0F52 0000 0F66 0000 0FA7 0000 0FE6 0000;
3AD1 1005 0000 104B 0000 10A2 0000 10E6 0000;
3AE1 112A 0000 1181 0000 11E1 0000 1241 0000;
3AF1 12CA 0000 1360 0000 1486 0000 1528 0000;

```

ASP FFT Program  
Program Validation Output

D.2.2 S-FFT, Impulse to DC Series

Impulse Time Input  
(1 of 128)

3A01	0000	0000	0000	0000	0000	0000	0000	0000;
3A11	0000	0000	0000	0000	0000	0000	0000	0000;
3A21	0000	0000	0000	0000	0000	0000	0000	0000;
3A31	0000	0000	0000	0000	0000	0000	0000	0000;
3A41	0000	0000	0000	0000	0000	0000	0000	0000;
3A51	0000	0000	0000	0000	0000	0000	0000	0000;
3A61	0000	0000	0000	0000	0000	0000	0000	0000;
3A71	0000	0000	0000	0000	0000	0000	0000	0000;
3A81	0000	0000	0000	0000	0000	0000	0000	0000;
3A91	0000	0000	0000	0000	0000	0000	0000	0000;
3AA1	0000	0000	0000	0000	0000	0000	0000	0000;
3AB1	0000	0000	0000	0000	0000	0000	0000	0000;
3AC1	0000	0000	0000	0000	0000	0000	0000	0000;
3AD1	0000	0000	0000	0000	0000	0000	0000	0000;
3AE1	0000	0000	0000	0000	0000	0000	0000	0000;
3AF1	0000	0000	0000	0000	0000	0000	0000	0000;

Frequency Output

3A01	2000	0000	2000	0000	2000	0000	2000	0000;
3A11	2000	0000	2000	0000	2000	0000	2000	0000;
3A21	2000	0000	2000	0000	2000	0000	2000	0000;
3A31	2000	0000	2000	0000	2000	0000	2000	0000;
3A41	2000	0000	2000	0000	2000	0000	2000	0000;
3A51	2000	0000	2000	0000	2000	0000	2000	0000;
3A61	2000	0000	2000	0000	2000	0000	2000	0000;
3A71	2000	0000	2000	0000	2000	0000	2000	0000;
3A81	2000	0000	2000	0000	2000	0000	2000	0000;
3A91	2000	0000	2000	0000	2000	0000	2000	0000;
3AA1	2000	0000	2000	0000	2000	0000	2000	0000;
3AB1	2000	0000	2000	0000	2000	0000	2000	0000;
3AC1	2000	0000	2000	0000	2000	0000	2000	0000;
3AD1	2000	0000	2000	0000	2000	0000	2000	0000;
3AE1	2000	0000	2000	0000	2000	0000	2000	0000;
3AF1	2000	0000	2000	0000	2000	0000	2000	0000;



ASP FFT Program  
Program Validation Output

(2 of 128) Time Input

```

3A01 0000 0000 0000 0000 0000 0000 0000 0000;
3A11 0000 0000 0000 0000 0000 0000 0000 0000;
3A21 0000 0000 0000 0000 0000 0000 0000 0000;
3A31 0000 0000 0000 0000 0000 0000 0000 0000;
3A41 0000 0000 0000 0000 0000 0000 0000 0000;
3A51 0000 0000 0000 0000 0000 0000 0000 0000;
3A61 0000 0000 0000 0000 0000 0000 0000 0000;
3A71 0000 0000 0000 0000 0000 0000 0000 0000;
3A81 0000 0000 0000 0000 0000 0000 0000 0000;
3A91 0000 0000 0000 0000 0000 0000 0000 0000;
3AA1 0000 0000 0000 0000 0000 0000 0000 0000;
3AB1 0000 0000 0000 0000 0000 0000 0000 0000;
3AC1 0000 0000 0000 0000 0000 0000 0000 0000;
3AD1 0000 0000 0000 0000 0000 0000 0000 0000;
3AE1 0000 0000 0000 0000 0000 0000 0000 0000;
3AF1 0000 0000 0000 0000 0000 0000 0000 0000;

```

Frequency Output

```

3A01 2000 0000 1FFB 0000 1FEC 0000 1FD3 0000;
3A11 1FB1 0000 1F85 0000 1F4F 0000 1F10 0000;
3A21 1EC8 0000 1E76 0000 1CD4 0000 1D0E 0000;
3A31 1D3E 0000 1D64 0000 1D7E 0000 1D8E 0000;
3A41 1D94 0000 1D8E 0000 1D7E 0000 1D64 0000;
3A51 1D3E 0000 1D0E 0000 1CD4 0000 1C90 0000;
3A61 1C42 0000 1BEA 0000 1B88 0000 1B1E 0000;
3A71 1AA8 0000 1A2C 0000 19A6 0000 1918 0000;
3A81 1880 0000 17E4 0000 173E 0000 1692 0000;
3A91 15E0 0000 1528 0000 146C 0000 13AA 0000;
3AA1 12E2 0000 1216 0000 1148 0000 1076 0000;
3AB1 0FA2 0000 0ECC 0000 0DF4 0000 0D1A 0000;
3AC1 0C40 0000 0B66 0000 0ABE 0000 09B6 0000;
3AD1 08DE 0000 080A 0000 0738 0000 06D7 0000;
3AE1 061F 0000 0563 0000 04A5 0000 03E3 0000;
3AF1 031F 0000 0259 0000 0191 0000 00C9 0000;

```

ASP FFT Program  
Program Validation Output

(4 of 128) Time Input

```

3A01 0000 0000 0000 0000 0000 0000 0000;
3A11 0000 0000 0000 0000 0000 0000 0000;
3A21 0000 0000 0000 0000 0000 0000 0000;
3A31 0000 0000 0000 0000 0000 0000 0000;
3A41 0000 0000 0000 0000 0000 0000 0000;
3A51 0000 0000 0000 0000 0000 0000 0000;
3A61 0000 0000 0000 0000 0000 0000 0000;
3A71 0000 0000 0000 0000 0000 0000 0000;
3A81 0000 0000 0000 0000 0000 0000 0000;
3A91 0000 0000 0000 0000 0000 0000 0000;
3AA1 0000 0000 0000 0000 0000 0000 0000;
3AB1 0000 0000 0000 0000 0000 0000 0000;
3AC1 0000 0000 0000 0000 0000 0000 0000;
3AD1 0000 0000 0000 0000 0000 0000 0000;
3AE1 0000 0000 0000 0000 0000 0000 0000;
3AF1 0000 0000 0000 0000 0000 0000 0000

```

Frequency Output

```

3A01 3FFC 0000 3FBD 0000 3EEB 0000 3D99 0000;
3A11 3BA8 0000 3D00 0000 3DC4 0000 3DFA 0000;
3A21 3D9C 0000 3CAE 0000 3B36 0000 3946 0000;
3A31 36E2 0000 3416 0000 30EC 0000 2D7E 0000;
3A41 29D4 0000 2602 0000 2218 0000 2189 0000;
3A51 1F33 0000 1C9D 0000 19D5 0000 16EF 0000;
3A61 13F5 0000 109A 0000 0E96 0000 0C60 0000;
3A71 09FC 0000 0782 0000 04F8 0000 0270 0000;
3A81 0002 0000 0250 0000 0482 0000 067C 0000;
3A91 0834 0000 09AA 0000 0AD6 0000 0BB4 0000;
3AA1 0D57 0000 0E85 0000 0F7B 0000 1035 0000;
3AB1 10AD 0000 10E3 0000 10D5 0000 10D6 0000;
3AC1 1152 0000 118A 0000 117E 0000 1132 0000;
3AD1 10A2 0000 0FD8 0000 0ED2 0000 0D9C 0000;
3AE1 0C3E 0000 0ABC 0000 0924 0000 0782 0000;
3AF1 05DC 0000 0489 0000 0313 0000 018D 0000

```

(8 of 128) Time Input

```

3A01 0000 0000 0000 0000 0000 0000 0000 0000;
3A11 0000 0000 0000 0000 0000 0000 0000 0000;
3A21 0000 0000 0000 0000 0000 0000 0000 0000;
3A31 0000 0000 0000 0000 0000 0000 0000 0000;
3A41 0000 0000 0000 0000 0000 0000 0000 0000;
3A51 0000 0000 0000 0000 0000 0000 0000 0000;
3A61 0000 0000 0000 0000 0000 0000 0000 0000;
3A71 0000 0000 0000 0000 0000 0000 0000 0000;
3A81 0000 0000 0000 0000 0000 0000 0000 0000;
3A91 0000 0000 0000 0000 0000 0000 0000 0000;
3AA1 0000 0000 0000 0000 0000 0000 0000 0000;
3AB1 0000 0000 0000 0000 0000 0000 0000 0000;
3AC1 0000 0000 0000 0000 0000 0000 0000 0000;
3AD1 0000 0000 0000 0000 0000 0000 0000 0000;
3AE1 0000 0000 0000 0000 0000 0000 0000 0000;
3AF1 0000 0000 0000 0000 0000 0000 0000 0000;

```

Frequency Output

```

3A01 3FF8 0000 3EAB 0000 3D14 0000 3F0C 0000;
3A11 3E32 0000 3AAC 0000 33BA 0000 2D6E 0000;
3A21 2830 0000 23C5 0000 1F75 0000 181A 0000;
3A31 1430 0000 0F34 0000 09CE 0000 048E 0000;
3A41 0000 0000 03C1 0000 0725 0000 09CC 0000;
3A51 0B72 0000 0DDE 0000 1284 0000 0FE0 0000;
3A61 0F48 0000 0DC4 0000 0C16 0000 0AC4 0000;
3A71 08DD 0000 0664 0000 04A0 0000 0266 0000;
3A81 0000 0000 024C 0000 0438 0000 0586 0000;
3A91 0747 0000 0869 0000 0C68 0000 09AE 0000;
3AA1 0A32 0000 0A08 0000 086E 0000 07D8 0000;
3AB1 061E 0000 04EA 0000 035E 0000 01A5 0000;
3AC1 0000 0000 01C2 0000 0382 0000 0508 0000;
3AD1 0622 0000 06AC 0000 0A58 0000 0809 0000;
3AE1 07FE 0000 07DC 0000 08B0 0000 0736 0000;
3AF1 061A 0000 04A0 0000 02FA 0000 0182 0000;

```

ASP FFT Program  
Program Validation Output

(16 of 128) Time Input

```

3A01 0800 0800 0800 0800 0800 0800 0800 0800;
3A11 0800 0800 0800 0800 0800 0800 0800 0800;
3A21 0000 0000 0000 0000 0000 0000 0000 0000;
3A31 0000 0000 0000 0000 0000 0000 0000 0000;
3A41 0000 0000 0000 0000 0000 0000 0000 0000;
3A51 0000 0000 0000 0000 0000 0000 0000 0000;
3A61 0000 0000 0000 0000 0000 0000 0000 0000;
3A71 0000 0000 0000 0000 0000 0000 0000 0000;
3A81 0000 0000 0000 0000 0000 0000 0000 0000;
3A91 0000 0000 0000 0000 0000 0000 0000 0000;
3AA1 0000 0000 0000 0000 0000 0000 0000 0000;
3AB1 0000 0000 0000 0000 0000 0000 0000 0000;
3AC1 0000 0000 0000 0000 0000 0000 0000 0000;
3AD1 0000 0000 0000 0000 0000 0000 0000 0000;
3AE1 0000 0000 0000 0000 0000 0000 0000 0000;
3AF1 0000 0000 0000 0000 0000 0000 0000 0000

```

Frequency Output

```

3A01 3FF4 0000 3DBC 0000 3EA0 0000 3412 0000;
3A11 2892 0000 1B1A 0000 14CE 0000 0970 0000;
3A21 0000 0000 06DF 0000 0B2C 0000 0D34 0000;
3A31 0D24 0000 0B06 0000 075C 0000 0494 0000;
3A41 0000 0000 03C4 0000 045A 0000 079C 0000;
3A51 08C6 0000 07AC 0000 053C 0000 0308 0000;
3A61 0000 0000 0296 0000 07A4 0000 07AC 0000;
3A71 06C6 0000 059C 0000 045E 0000 023A 0000;
3A81 0002 0000 0222 0000 071E 0000 0696 0000;
3A91 058E 0000 04B4 0000 0356 0000 01CF 0000;
3AA1 0000 0000 01F0 0000 039C 0000 0515 0000;
3AB1 04B0 0000 04DC 0000 03CE 0000 01A7 0000;
3AC1 0000 0000 01C6 0000 0130 0000 02F4 0000;
3AD1 03F8 0000 050E 0000 03AC 0000 0185 0000;
3AE1 0000 0000 01A6 0000 0350 0000 0263 0000;
3AF1 0400 0000 03F3 0000 02C6 0000 017A 0000

```



ASP FFT Program  
Program Validation Output

(32 of 128) Time Input

```

3A01 0000 0000 0000 0000 0000 0000 0000 0000;
3A11 0000 0000 0000 0000 0000 0000 0000 0000;
3A21 0000 0000 0000 0000 0000 0000 0000 0000;
3A31 0000 0000 0000 0000 0000 0000 0000 0000;
3A41 0000 0000 0000 0000 0000 0000 0000 0000;
3A51 0000 0000 0000 0000 0000 0000 0000 0000;
3A61 0000 0000 0000 0000 0000 0000 0000 0000;
3A71 0000 0000 0000 0000 0000 0000 0000 0000;
3A81 0000 0000 0000 0000 0000 0000 0000 0000;
3A91 0000 0000 0000 0000 0000 0000 0000 0000;
3AA1 0000 0000 0000 0000 0000 0000 0000 0000;
3AB1 0000 0000 0000 0000 0000 0000 0000 0000;
3AC1 0000 0000 0000 0000 0000 0000 0000 0000;
3AD1 0000 0000 0000 0000 0000 0000 0000 0000;
3AE1 0000 0000 0000 0000 0000 0000 0000 0000;
3AF1 0000 0000 0000 0000 0000 0000 0000 0000

```

Frequency Output

```

3A01 3FF0 0000 3E5C 0000 28AA 0000 148A 0000;
3A11 0000 0000 0C68 0000 0D71 0000 08CC 0000;
3A21 0000 0000 06CC 0000 07A0 0000 0584 0000;
3A31 0000 0000 04A0 0000 05CA 0000 03F4 0000;
3A41 0000 0000 036E 0000 04C0 0000 0304 0000;
3A51 0000 0000 02AE 0000 0412 0000 0290 0000;
3A61 0000 0000 026B 0000 0396 0000 0246 0000;
3A71 0000 0000 0229 0000 0336 0000 0208 0000;
3A81 0000 0000 01F0 0000 02E6 0000 01D9 0000;
3A91 0000 0000 01C5 0000 02A4 0000 01B3 0000;
3AA1 0000 0000 01A0 0000 026A 0000 0184 0000;
3AB1 0000 0000 0188 0000 023E 0000 0182 0000;
3AC1 0000 0000 018A 0000 020A 0000 0184 0000;
3AD1 0000 0000 0188 0000 01FF 0000 0186 0000;
3AE1 0000 0000 018A 0000 01FE 0000 0184 0000;
3AF1 0000 0000 018A 0000 0202 0000 0180 0000

```

ASP FFT Program  
Program Validation Output

(64 of 128) Time Input

```

3A01 0800 0800 0800 0800 0800 0800 0800 0800;
3A11 0800 0800 0800 0800 0800 0800 0800 0800;
3A21 0800 0800 0800 0800 0800 0800 0800 0800;
3A31 0800 0800 0800 0800 0800 0800 0800 0800;
3A41 0800 0800 0800 0800 0800 0800 0800 0800;
3A51 0800 0800 0800 0800 0800 0800 0800 0800;
3A61 0800 0800 0800 0800 0800 0800 0800 0800;
3A71 0800 0800 0800 0800 0800 0800 0800 0800;
3A81 0000 0000 0000 0000 0000 0000 0000 0000;
3A91 0000 0000 0000 0000 0000 0000 0000 0000;
3AA1 0000 0000 0000 0000 0000 0000 0000 0000;
3AB1 0000 0000 0000 0000 0000 0000 0000 0000;
3AC1 0000 0000 0000 0000 0000 0000 0000 0000;
3AD1 0000 0000 0000 0000 0000 0000 0000 0000;
3AE1 0000 0000 0000 0000 0000 0000 0000 0000;
3AF1 0000 0000 0000 0000 0000 0000 0000 0000

```

Frequency Output

```

3A01 3FF4 0000 1E32 0000 0001 0000 0C34 0000;
3A11 0000 0000 070E 0000 0004 0000 0511 0000;
3A21 0000 0000 0330 0000 0002 0000 03D4 0000;
3A31 0000 0000 035C 0000 0002 0000 0466 0000;
3A41 0000 0000 0400 0000 0002 0000 0362 0000;
3A51 0000 0000 02C8 0000 0000 0000 01AA 0000;
3A61 0000 0000 01D0 0000 0002 0000 02CB 0000;
3A71 0000 0000 0360 0000 0000 0000 0368 0000;
3A81 0000 0000 02E1 0000 0002 0000 01A7 0000;
3A91 0000 0000 011E 0000 0001 0000 011C 0000;
3AA1 0000 0000 01C6 0000 0002 0000 0242 0000;
3AB1 0000 0000 0275 0000 0001 0000 01E4 0000;
3AC1 0000 0000 01A6 0000 0002 0000 0060 0000;
3AD1 0000 0000 002C 0000 0000 0000 013A 0000;
3AE1 0000 0000 0171 0000 0004 0000 0156 0000;
3AF1 0000 0000 016D 0000 0000 0000 00E6 0000

```

ASP FFT Program  
Program Validation Output

DC Time Input

```

3A01 0800 0800 0800 0800 0800 0800 0800 0800;
3A11 0800 0800 0800 0800 0800 0800 0800 0800;
3A21 0800 0800 0800 0800 0800 0800 0800 0800;
3A31 0800 0800 0800 0800 0800 0800 0800 0800;
3A41 0800 0800 0800 0800 0800 0800 0800 0800;
3A51 0800 0800 0800 0800 0800 0800 0800 0800;
3A61 0800 0800 0800 0800 0800 0800 0800 0800;
3A71 0800 0800 0800 0800 0800 0800 0800 0800;
3A81 0800 0800 0800 0800 0800 0800 0800 0800;
3A91 0800 0800 0800 0800 0800 0800 0800 0800;
3AA1 0800 0800 0800 0800 0800 0800 0800 0800;
3AB1 0800 0800 0800 0800 0800 0800 0800 0800;
3AC1 0800 0800 0800 0800 0800 0800 0800 0800;
3AD1 0800 0800 0800 0800 0800 0800 0800 0800;
3AE1 0800 0800 0800 0800 0800 0800 0800 0800;
3AF1 0800 0800 0800 0800 0800 0800 0800 0800

```

Frequency Output

```

3A01 3A2E 0000 05BC 0000 04CE 0000 035E 0000;
3A11 01F6 0000 005E 0000 00C0 0000 0130 0000;
3A21 00EB 0000 005C 0000 0094 0000 00DE 0000;
3A31 00C6 0000 0055 0000 0088 0000 0102 0000;
3A41 0109 0000 00D0 0000 0068 0000 0082 0000;
3A51 00B8 0000 0086 0000 0038 0000 004C 0000;
3A61 0066 0000 004C 0000 0075 0000 00E5 0000;
3A71 0142 0000 013A 0000 0106 0000 013A 0000;
3A81 017F 0000 01BC 0000 0192 0000 0142 0000;
3A91 0147 0000 0165 0000 016A 0000 0120 0000;
3AA1 00CA 0000 00B6 0000 00BE 0000 009C 0000;
3AB1 004C 0000 001F 0000 004D 0000 0046 0000;
3AC1 002D 0000 0053 0000 0086 0000 00AE 0000;
3AD1 008E 0000 007A 0000 0084 0000 00C6 0000;
3AE1 00C0 0000 00BC 0000 00B4 0000 00B0 0000;
3AF1 00C0 0000 00E0 0000 0102 0000 00C2 0000

```

16 4339-Q













GENERAL LIBRARY - U.C. BERKELEY

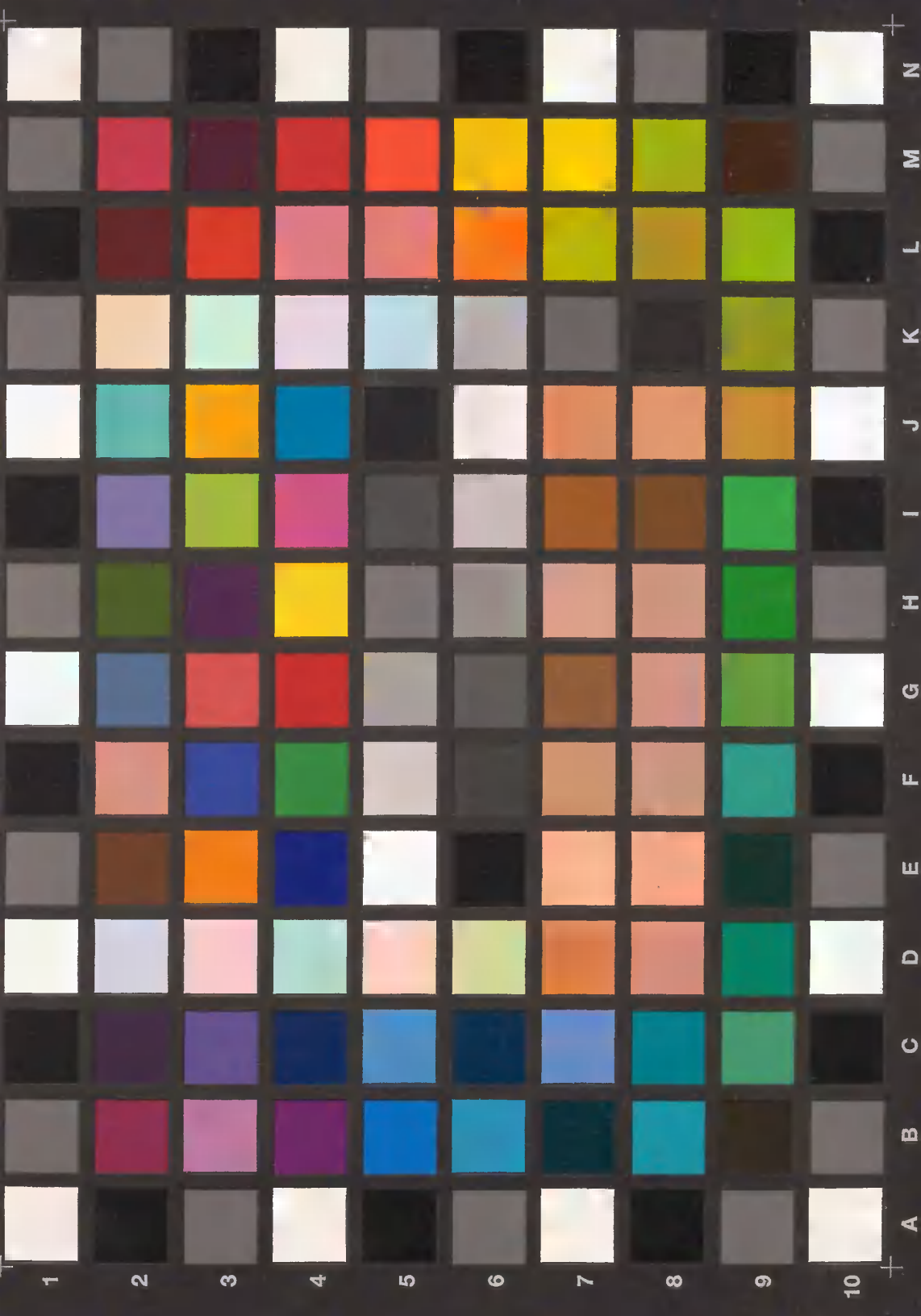


8000132609

U.C. BERKELEY  
ENGINEERING LIBRARY







+

N

M

L

K

J

I

H

G

F

E

D

C

B

A

+

1

2

3

4

5

6

7

8

9

10







# THE ARCHIVE

